

Contribution to Temporal Fault Tree Analysis without Modularization and Transformation into the State Space

Translation into English
of the doctoral thesis of
Dr. Ing. Simon J. Schilling
at the
Bergische Universität Wuppertal.

Date of examination:
21. December 2009

Reviewer/Supervisor:
Univ.-Prof. Dr.-Ing. A. Meyna
Univ.-Prof. Dr. rer.nat. P. C. Müller

The german original can be downloaded from
<http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:468-20100070>
Translated version of 19. Mai 2015.

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
See inside for more details.

For
Albert and Alexandra and Liselotte

Preface to the Translation

This translation into English was done in order to present my work to a broader audience. I aimed at staying as close to the german original as possible. This is especially relevant for the state of the art chapter which was not updated. Thus, newer work, as well as additional work by authors that were already referenced in the original, was not taken into account.

The german original is an official doctoral (i.e. Ph.D.) thesis and was published and is hosted as PDF by the university itself. I chose to publish this translation – including the complete latex sources – under a CreativeCommons license and host it at github because I was looking for a simple, stable and open – as in open source – solution for the benefit of potential readers. As English is not my first language, I surely made some mistakes and would greatly appreciate any comments and suggestions for improvements.

Munich, May 2015

Simon Schilling

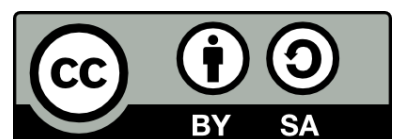
License

“Contribution to Temporal Fault Tree Analysis without Modularization and Transformation into the State Space” by Simon J. Schilling is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>.

It is based on the work at <http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:468-20100070>, which is the german original version of this thesis, and also by Simon J. Schilling.

Note, that the german original is not published under a Creative Commons License.



Preface

This work was accomplished during my time as scientific member of the Central Functional Safety Team at BMW Group in Munich, Germany.

I want to specifically thank Univ.-Prof. Dr.-Ing. Arno Meyna and Dipl.-Ing. Christoph Jung.

I thank Professor Meyna, for his support during my external promotion at the department of safety engineering, safety theory and traffic engineering at the Bergische Universität Wuppertal.

I thank Mr. Jung, who was head of the Central Functional Safety Team at BMW Group and convenor of ISO TC 22 SC 3 WG 16 and as such one of the main creative heads behind and responsible for ISO 26262, for making this work possible and I thank him for repeatedly trusting and supporting me throughout the last years.

I thank Prof. Dr. rer. nat. P. C. Müller for writing the second assessment on this work and being part of the graduation committee. I thank Prof. Dr.-Ing. Dipl.-Wirtsch.-Ing. B. H. Müller for chairing the graduation committee. I thank Prof. Dr.-Ing. U. Barth for being part of the graduation committee.

I thank my colleagues at BMW for their support and interest.

I especially thank Dr.-Ing. Martin Woltereck, who brought me to the field of functional safety and to fault tree analysis.

Munich, December 2009

Simon Schilling

Abstract

Background

Fault tree analysis (FTA) is a well established method for qualitative as well as probabilistic reliability and safety analysis. Fault trees are Boolean models and thus do not support modelling of dynamic effects like sequence dependencies between fault events. In order to overcome this limitations, *dynamic fault tree* methods were defined previously. Most of these are based on complete or partial transformation of the fault tree model into state-space-models like Markov chains or Petri nets. These state-space-models generally suffer from exponential state explosion which imposes the necessity to define small “dynamic” modules which need to be independent from the rest of the model. Moreover, these state-space-models lack some of the FTA’s benefits like logical simplification of complex system functions or a real cutset analysis. Because of these deficiencies, a method is needed that allows consideration of sequence dependencies without transformations into state-space. This work describes such a new approach.

Concept

The new *temporal fault tree analysis* (TFTA) described in this work extends the Boolean FTA in order to take sequence dependencies into account. The TFTA is based on a new *temporal logic* which adds a *concept of time* to the Boolean logic and algebra. This allows modelling of temporal relationships between events using Boolean operators (AND “ \wedge ”, OR “ \vee ”, NOT “ \neg ”) and two new temporal operators (PAND “ $\overline{\wedge}$ ” and SAND “ $\overline{\vee}$ ”). With a set of *temporal logic rules*, a given *temporal term* may be simplified to its *temporal disjunctive normal form* (TDNF) which is similar to the Boolean DNF but includes event sequences. In TDNF the top event’s temporal system function may be reduced to a list of *minimal cutset sequences* (MCSS). These allow qualitative analyses similar to Boolean cutset analysis in normal FTA. Furthermore the TFTA may also be used for probabilistic analyses. Probabilities and rates of MCSS may be calculated without using state-space models. Again the procedure is similar to the normal FTA: top event failure probabilities and rates are derived from the failure probabilities and rates of the basic events including sequence dependencies.

Realisation

Starting with the Boolean FTA this work describes a new notation and new rules for a temporal logic. This temporal logic aims at transforming temporal terms into a TDNF, which then may be transformed further into a form where all terms are mutually exclusive. This form is well suited for quantification, too. Several examples are provided which explain each step in detail. Furthermore, there are two probabilistic approximation methods described, which allow a significant reduction of the calculatory effort.

Results

One significant aspect of the new TFTA described in this work is the possibility to take sequence dependencies into account for qualitative and probabilistic analyses without state-space transformations. Among others, this allows for modelling of event sequences at all levels within a fault tree, a real qualitative analysis similar to the FTA’s cutset analysis, and quantification of sequence dependencies within the same model.

General Remark and Disclaimer

All safety and reliability analyses in this work are presented solely for the purpose of demonstrating new analysis methods and are to be seen as simplifications and examples only. While they use, among others, technical functions and data similar to those of real systems, they must not be taken as evidence for the safety or reliability of existing or planned “real life” systems, functions, or components.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Structure of this Thesis	2
2	State of the Art: Static and Dynamic Fault Tree Analysis (FTA)	3
2.1	Background	3
2.1.1	Reliability and Safety Analyses	3
2.1.2	Static and Dynamic Analyses	4
2.1.2.1	Dynamic System Behaviour	4
2.1.2.2	Methods of Modelling	4
2.2	Static FTA – the Classical Approach	5
2.3	Dynamic FTA	6
2.3.1	Defining Dynamic with Event Sequences	6
2.3.2	Other Definitions of Dynamic	7
2.3.3	Dynamic FTA – Other Approaches	8
2.3.4	Dynamic FTA – Based on a Temporal Failure Logic	10
2.4	Summary	13
3	Problem Definition: Event Sequences in FTA without Modularization	15
3.1	Demand for Improved Methods	15
3.1.1	Demand for Dynamic FTA	15
3.1.2	Demand for Improved Dynamic FTA	15
3.1.3	Remarks on Using Dynamic FTA	16
3.2	Concept	16
3.2.1	Requirements for TFTA	16
3.2.2	TFTA – Step by Step	17
4	Temporal Fault Tree Analysis (TFTA): A New Approach to Dynamic FTA	19
4.1	TFTA Notation	19
4.1.1	Boolean Algebra and the FTA Failure Logic	19
4.1.2	Temporal Logic Operators	20
4.1.3	Boolean and Temporal Operations Visualized as Sets	21
4.1.4	Temporal Operations: Timing Behaviour	22
4.1.5	Syntax of Temporal Expressions	23
4.1.5.1	Temporal Disjunctive Normal Form (TDNF, Sum of Products)	25
4.1.5.2	Extended TDNF (Sum of Products)	26
4.1.6	Events the are “Part” of an Expression	27
4.1.7	Visualization Using Sequential Failure Trees	27
4.1.7.1	Normal Sequential Failure Trees (without SAND)	28
4.1.7.2	Sequential Failure Trees with Concurrent Events/SAND	29
4.1.7.3	Using Sequential Failure Trees	29

4.2	Laws of the TFTA Temporal Logic	30
4.2.1	Boolean Algebra	31
4.2.2	Law of Completion	31
4.2.3	Law of Contradiction	32
4.2.4	Temporal Law of Idempotency	32
4.2.5	Temporal Law of Commutativity	32
4.2.6	Temporal Law of Associativity	33
4.2.7	Further Temporal Logic Laws	33
4.2.8	Temporal Operations with Negated Events	37
4.2.8.1	How to Interpret Negated Events in TFTA	37
4.2.8.2	Using Negated Events in TFTA	37
4.2.8.3	Rules of Replacement for Negated Events in the Temporal Logic	38
4.2.8.4	Conjunction of Negated Events	40
4.2.8.5	Temporal Laws of Negation	40
4.2.9	<i>True</i> and <i>False</i> in Temporal Logics	41
4.2.10	Temporal Distributive Laws	41
4.2.10.1	Distributive Law for PAND-OR Expressions of Type I	41
4.2.10.2	Distributive Law for PAND-OR Expressions of Type II	43
4.2.10.3	Distributive Law for SAND-OR Expressions	44
4.2.11	Temporal Laws of Absorption	45
4.2.12	Temporal Law for Intersections	47
4.3	Minimal and Disjoint Forms of TFTA Temporal Expressions	47
4.3.1	Minimal and Disjoint Forms of Boolean Expressions	47
4.3.2	Minimal Temporal Expressions	48
4.3.2.1	Structurally Non-Minimal Temporal Expressions	49
4.3.2.2	Temporally Non-Minimal Temporal Expressions	49
4.3.3	Disjoint Temporal Expressions	51
4.3.3.1	Condition for Disjointness	51
4.3.3.2	Structurally and Temporally Disjoint Temporal Expressions	51
4.3.3.3	Disjoint Separation Using Temporal Minterms	52
4.4	Simplification Using Extended Event Sequences	52
4.4.1	Motivation and Requirements	53
4.4.2	Using Extended Temporal Expressions	54
4.5	Summary	55
5	Probabilistic Quantification of the TFTA Method	57
5.1	Quantification of the Boolean FTA	57
5.2	Quantification of the TFTA: Temporal Concept and Failure Frequencies	59
5.2.1	Sequences with Two Events	59
5.2.2	Sequences with More Than Two Events	60
5.2.3	What Parameter to Use in Probabilistic Analyses?	61
5.3	Quantification of the PAND and SAND Operators	61
5.3.1	Quantification Using Logic Functions	61
5.3.2	Quantification Using Comparison with State Diagrams	63
5.4	Quantification of the Temporal Failure Function	66
5.4.1	Quantification of Event Sequences and MCSS	66
5.4.2	Quantification of Extended Event Sequences	68
5.4.3	Quantification of the Temporal Failure Function on TOP Level	69

5.5	Reducing the Computing Time	70
5.5.1	Temporal Terms in MCSS Format	70
5.5.2	Temporal Terms in an Extended MCSS Format	72
6	Comparing TFTA to Other Dynamic Modelling Approaches	75
6.1	An Example System	75
6.2	Comparison with the Boolean FTA	76
6.3	Comparison with Dynamic FTA (DFT Method)	78
6.4	Comparison with Markov Diagrams	79
6.5	Dynamic FTA According to the TFTA Method	80
6.6	Summarizing the Results	82
7	TFTA Analysis of an Automotive ECU Architecture	85
7.1	The Example System	85
7.1.1	System Description, Safety Goal and Safe State	86
7.1.2	Failures	87
7.2	Temporal Fault Tree	87
7.3	Qualitative Analysis of the Temporal Fault Tree	91
7.3.1	Temporal Failure Function	91
7.3.2	Transformation According to the Temporal Logic Rules	91
7.3.3	Analysis of the MCSS	98
7.4	Probabilistic Analysis of the TOP Failure Parameters	101
7.5	Discussion	102
8	Summary and Outlook	105
9	Bibliography	109
	Appendix	114
A	Further Explanations on Selected Topics	117
A.1	Reliability Characteristics	117
A.2	Creating and Using Sequential Failure Trees in the TFTA	118
A.3	Examples: Mutually Exclusive (Disjoint) Temporal Expressions	120
B	Abbreviations/Acronyms	125
C	Notation	127

1 Introduction

System safety is organized common sense.

(Mueller)

1.1 Motivation

The fault tree analysis (FTA) is one of the most important methods of modelling and analyzing the reliability and safety of systems qualitatively as well as probabilistically. In the automotive domain there is a trend towards more safety critical electronics [1], and thus functional safety is increasingly important [2]. Therefore the domain specific functional safety standard ISO 26262 [3] is currently being derived from the more generic IEC 61508 [4].

In the automotive domain the FTA is used during development for several reasons: the allocation of safety requirements, as well as the confirmation and verification of requirements (e.g. failure rates as required by ISO 26262), and the comparison of safety architectures.

Today, the FTA is generally considered as state of the art, e.g. [5–9]. Nevertheless certain problems remain, and there is an ongoing scientific interest for the FTA method.

This thesis results from years of practise experience during my time at the functional safety department of a german automotive manufacturer. Contrary to expectations, the conventional – i.e. static – FTA is still having difficulties at providing realistic and not too conservative results when applied to modern electric/electronic (EE) systems.

The operational behaviour and failures of such systems are highly dynamic in a sense that subsystems, functions and components (or their failures) depend on each other (structural dependencies) or depend on their relative timing (temporal dependencies) [10].

The fault tree methode on the other hand is limited to binary parameters as it is based on Boolean (failure-)logic. As a consequence, temporal dependencies and dependencies between failure rates of fault tree basic events must be omitted. Both limitations may usually be circumvented, or at least mitigated, by taking specific assumptions and approximations into account. But both problems can not be completely solved from within the conventional FTA.

Furthermore, when using fault trees one has to keep in mind that conservative approximations (less modelling effort) usually conflict with the wish to avoid an unnecessarily expensive system design. Unprecise (approximated) fault tree models must not lead to overly complex and overly expensive technical solutions in the system under consideration.

This problem and conflict is well known [11–13]. In general, there is always the possibility to analyze the system using other methods that can take dynamic effects into account, like e.g. state based methods. On the other hand there is a reason for the FTA's success as one of the most widely used methods for analyzing the reliability and safety of complex systems [14]: in

comparison to other methods fault trees are easy to use, to read, to understand, and they are scalable. This is, because a system's fault tree is similarly structured as the system architecture. Especially state based methods (e.g. markov diagrams) lack this feature.

For years there have been several approaches to combine state based methods with the conventional FTA. These aim at combining the benefits of both methods while circumventing their disadvantages. Usually the user shall stay within the more intuitive fault tree, while modelling the system under consideration; then, the system's dynamic effects and dependencies are hidden from the user by state based models that do the calculations in the background automatically.

Such hybrid techniques are often called *dynamic FTA*; but they also have some specific disadvantages. Mostly they use fault trees as a tool for easy visualization or relatively simple creation of models; but they do not also fully use the fault tree for the analysis and calculation, and thus they do without some of the FTA's biggest benefits.

These problems, as well as pure scientific curiosity, lead to intense research on a more efficient way to handle dynamic effects and dependencies from within fault trees.

This thesis presents the results of this research.

1.2 Structure of this Thesis

This thesis deals with dynamic effects in safety and reliability analyses, and specifically with the modelling of failure sequences in fault trees. It is structured as follows.

Chapter 2 presents the state of the art as relevant for this thesis; specific focus goes to the conventional Boolean FTA (chapter 2.2), as well as to dynamic extensions of the FTA (chapter 2.3); the latter includes methods where the fault tree model is transformed into a state based model, as well as methods using temporal logics.

This survey points to several shortcomings of the current state of the art; specifically these result from changing the modelling and analysis and calculation's focus and are listed in chapter 3 which also derives criteria and requirements for improvements.

Chapter 4 describes the proposed new approach for including failure event sequences into the fault tree without changing to the state space. This new *temporal fault tree analysis* (TFTA) relies on an temporal extension to the conventional Boolean algebra and logic; this *temporal logic* has its own notation (chapter 4.1) and its own laws of transformation (chapter 4.2). Chapter 4.3 then shows how to transform temporal terms into disjunct minimal failure event sequences. There is also an extended form of the TFTA which is presented in chapter 4.4; it allows for reduced calculatory effort when solving more complex temporal failure functions.

Chapter 5 discusses the quantification of temporal terms, which in turn allows probabilistic evaluation of temporal fault trees.

Chapter 6 compares the new TFTA approach with a) conventional Boolean FTA, b) the dynamic fault tree approach (DFT) as a typical dynamic extension of the Boolean FTA, and c) markov diagrams.

Chapter 7 applies the TFTA to a more complex and complete example in order to demonstrate its practical use. A typical automotive ECU architecture is analyzed: beginning with its system analysis, followed by creation of a corresponding temporal fault tree, and finally the qualitative as well as probabilistic fault tree transformation and analysis.

This thesis closes with a summary and outlook in chapter 8.

2 State of the Art: Static and Dynamic Fault Tree Analysis (FTA)

Of course, it is safe, we certified it.

(An FAA administrator)

This chapter provides an overview over the state of the art as relevant for the TFTA method.

- Chapter 2.1 describes the field of safety related fault tree analysis in general.
- The conventional and solely static FTA is among the most common methods for systematic top down failure analysis of complex systems, see chapter 2.2.
- As shown in chapter 2.3, today there are several extensions to the conventional FTA; they take dynamic failure behaviour into account and try to mitigate the FTA's shortcomings in this field. Chapter 2.3.3 presents state based methods, and methods using temporal (failure) logics are discussed in chapter 2.3.4.
- Chapter 2.4 summarizes the state of the art, which leads to the main problem description of this thesis in the following chapter 3.

2.1 Background

2.1.1 Reliability and Safety Analyses

The *reliability* of a system or a component (in general: an entity) is defined as its “capability [...] to meet expected performance criteria, given by its intended use, during a defined time period [15]. An entity that has failed can no longer provide its functionality; therefore, conventional *reliability analysis* reflects upon entities’ failure behaviour.

Such an analysis usually covers the following steps [16]: it supports development of new systems by comparing different – existing or proposed – system designs among each other, as well as comparing them to objective requirements (i.e. *reliability prediction*, *reliability comparison*, *reliability pursuit*, *identification of weak spots*). Additionally, it allows *reliability verification* of existing systems and concepts. The same methods and analytical approaches are usually used for all these purposes.

In comparison to reliability analysis, the *safety analysis* is focused on only those system and component failures that lead to loss of “safety”, where safety is defined as “freedom from unacceptable risks” [4]. From a safety perspective, an entity’s relevant reliability is therefore its

capability – or, in case of a more probabilistic view, its probability – to not induce *dangerous effects* (i.e. damage) during a defined time period and under given circumstances. Thus, reliability, from a safety perspective, takes failures consequences into account, too.

Safety analyses therefore need to define which risks and which damages are relevant. In the context of conventional safety of technical systems these typically are the danger for life and limbs, or injuries and death of persons [4]. In general, the same analysis methods are used in other contexts, too; e.g. in the context of security of technical systems [17, 18]. This thesis only addresses the safety context¹.

2.1.2 Static and Dynamic Analyses

2.1.2.1 Dynamic System Behaviour

A system behaves dynamically if [19] the system response to a initial disturbance develops over time, while the system’s components interact among each other, as well as with their surrounding. In comparison, conventional fault tree analysis looks at unwanted events (i.e. system failures) as static, determined, and time invariant consequence to certain component failures [19].

In a world full of dynamic influences and interactions basically all technical systems also behave dynamically. Statistical methods and models for reliability and safety analysis of systems therefore necessarily only approximate a system’s real dynamic behaviour.

This simplification is the main reason why handling of statistical analysis like FTA or *reliability block diagrams* (RBD) is relatively easy. Actually, in many cases it is the assumption of static behaviour that makes an analysis feasible at all. In practise the relevant question is which static approximations allow “good enough” representation of the actual dynamic failure behaviour.

It has been demonstrated that conventional FTA is very well suited for logical and probabilistic analyses of systems, if their failure behaviour is – at least in the first approximation – free of time dependencies or dynamic interactions between its components.

On the other hand, and since the very beginning of systematic failure behaviour analysis after the mid-20th century, researchers and users are complaining about static analysis being too imprecise [20]. Therefore, scientists are researching how static analysis methods like FTA may be extended by the most important dynamic effects – but without excessively increasing modelling and calculatory effort.

2.1.2.2 Methods of Modelling

In sight of [21] and [22] three types of *dynamic reliability and safety analyses* (ZSA) may be distinguished by their different modelling approaches. These are

- state transition models, especially markov models, e.g. [23],
- direct simulation of systems, especially using MoCaS, e.g. [13, 24], and
- extensions of static event sequence analysis and the FTA in order to also represent dynamic effects.

The following chapters cover those methods in more detail.

¹Author’s remark: in german there is only one term “Sicherheit” for both of the english “safety” and “security”; therefore, a further distinction and limitation of this thesis’ scope follows at this place, but is omitted in the english translation.

2.2 Static FTA – the Classical Approach

The history of FTA can be traced to the mid-20th century and starts with the reliability analysis of the Minuteman missile [25, 26].

The conventional fault tree [6–8] is a Boolean model, that systematically and methodically describes the interaction of failures within a system that lead to a system failure. It is a top down or deductive method. Starting from an undesirable event or system state – the so-called *TOP* –, more detailed failure events are searched for iteratively, that cause this TOP. Graphical representation of these failure events is done using a tree notation, the so-called *fault tree*. The components' failure events modelled in the fault tree are represented by events that can be in one of two states according to Boolean logic: “intact/unfailed/failure has not occurred” is represented by a Boolean *False* or 0, and “defect/failed/failure has occurred” is represented by a Boolean *True* or 1, respectively.

Evaluation of the fault tree is done qualitatively as well as probabilistically. The system is comprised of clearly separable elements (components), each of which has its own reliability and safety characteristics, and that influence the system reliability and safety according to the components' logical interconnection. Using these connections, the fault tree model is then able to derive the system characteristics from its component characteristics.

With the simplifying laws of Boolean algebra the *system function/failure function*, i.e. the logical function of the TOP event, is transformed into a minimal disjunctive normal form. Thereby determined *minimal cutsets* of the fault tree may then be further used probabilistically together with the laws of probability calculus. The probability or frequency of occurrence of the undesirable event or system state is – for non-repairable systems – the failure probability and the failure density or failure rate of the TOP event, respectively; for repairable systems, it is the unavailability and failure frequency of the TOP event, respectively, [27].

Furthermore, qualitative analysis of the system architecture is possible, too, because of the similarity of the fault tree model to the real system structure; specifically, such qualitative analysis allows analysis of redundancy structures as well as sensitivity analysis [28], importance analysis [29], and confidence analysis [30].

Qualitative and probabilistic static FTA is state of the art in many domains like nuclear [5], aerospace [31], and automotive industries [9, 32]. There is demand for further research on using FTA for analysis of software “failures” [33], especially because of difficulties stemming from proper representation of dynamic effects, see below.

FTA is intuitive in its application – in comparison to other methods like e.g. state based markov diagrams; thus, learning the FTA method is comparatively easy, and fault trees are easy to create, read, understand, rework, and edit, as well as to detail iteratively, and to use in modules.

One main limitation of the FTA is that its event are (only) bivalent, i.e. *True* or *False*; another limitation is that the *assumptions of monotony* or *coherence* must be satisfied [34, 35]; a third limitation is the implied independence of its basic events. Furthermore, FTA has only very limited possibilities of representing dynamic failure and repair behaviour [12]. Reason for this is the underlying Boolean logic [36], that has no concept of time, and thus only covers structural aspects of failure combinations [34]. No statement is made about the sequence in which events occur, as well as about other time dependencies, see chapter 2.3.1.

2.3 Dynamic FTA

The expression *dynamic FTA* is often used as a synonym for the *dynamic fault tree* (DFT) approach according to Dugan [37]. The DFT uses markov chains to extend the static FTA to model and to analyze *sequence dependencies*.

The DFT approach therefore defines its “dynamic” with event sequences. This thesis and the TFTA approach, as described in chapter 4, are also based on this underlying interpretation of “dynamic”, i.e. on the possibility of representing event sequences.

2.3.1 Defining Dynamic with Event Sequences

Boolean logic with its AND, OR, and NOT operations is not capable of expressing temporal relationships. For example, the failures of two components *A* and *B* in a system shall be considered. The event “*A* AND *B*” represents “both components have failed”. It does not, though, provide any information on the real points in time at which *A* and *B* occurred, and from that: the sequence, in which both events occur. This Boolean view grasps only the static state that the two components are (or are not) failed.

In contrast to that, a dynamic view discriminates between different ways of reaching this event or state. It extends the all-static analysis of only considering possible combinations of events [38].

For “*A* AND *B*” there are three different such ways. First, *A* may fail before *B*, and then *B* fails later, too. Second, *B* may fail before *A*, and then *A* fails later, too. Third, *A* and *B* may fail exactly simultaneously.

Each of these ways leads to the – from a Boolean point of view: identical – state, that both components have failed. This discrimination of possible ways to an event or state may be visualized using state-transition diagrams. Figure 2.1 shown such a state-transition diagram, corresponding to the example above. “Dynamic” as discrimination of different ways to an event or state works with temporal expressions like “before”, “after”, “first”, “then”, “simultaneous”, and so on. Modelling such “dynamics” requires to differentiate the different points in time when events occur. This capability requires that a *concept of time* exists within the model [39].

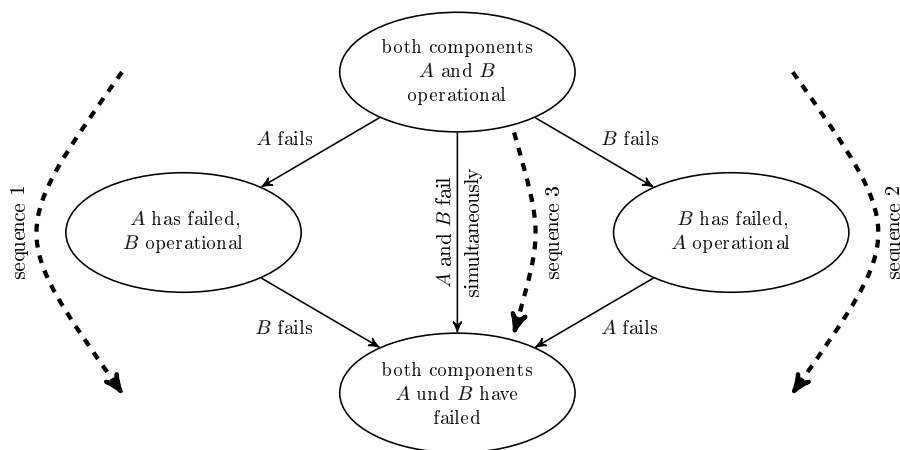


Figure 2.1: State-transition diagram of a simple, redundant, and non-reparable system, that consists of two components *A* and *B*; this diagram shows the possible four states and five transitions.

Conversely, differentiating points in time when events occur also allows to distinguish between different event sequences. And with event sequences a multitude of dynamic effects can be described [12, 40].

Next Steps

The contribution to dynamic FTA, as presented in this thesis, also uses “dynamic” in the sense of representation of event sequences. The next section 2.3.2 differentiates this meaning of “dynamic” from others that are also used in the context of ZSA, and specifically are used in the context of FTA. Section 2.3.3 discusses typical implementations of this meaning of “dynamic”, specifically implementations based on markov chains and petri nets. Section 2.3.4 outlines a very different way of extending the FTA by event sequences, and for this purpose describes several approaches of extended (temporal) failure logics. Chapter 2.4 summarizes this state of the art of dynamic FTA.

2.3.2 Other Definitions of Dynamic

Apart from the consideration of event sequences there are other temporal dependencies among (failure) events, and consequently other definitions of “dynamic” in the ZSA field, too, some of which are listed below. One overview in [41] is not the most recent, but is still valid.

In [42] dynamic effects in analyses result either from time-dependent failure rates, or from time-dependent unavailabilities, or from reduction of uncertainty whether the reliability data used is correct, or from failure sequences.

Abstracting these categories, dynamic either results from variable reliability data, or from the failure events’ sequence. Sometimes, *phased mission* methods are seen as a third such category, see e.g. [39, 43] or [44]. But these may as well be seen as belonging to either of the first two categories, or they may be interpreted as piecewise static analysis.

A further distinction into “fast” and “slow” dynamic temporal dependencies is given in [22]. Slow dynamic effects occur during normal operation, e.g. by aging, learning effects, or changes in the system. On the other hand, fast dynamic effects describe incidents, and thus dynamic ZSA focus on these. In [22] dynamic ZSA is based on MoCaS.

The referenced work comes mainly from the nuclear domain. They emphasize explicit consideration of temporal dependencies as well as consideration of *HRA* (HRA) [45] as another important contribution of dynamic ZSA. On the other hand, HRA is not as relevant in the automotive domain today; reasons for this are

1. that safety critical systems are preferably designed as *fail safe systems*, thus real *fail operational systems* are rare [46],
2. the lack of human operators as part of the safety systems, which directly influence the system’s behaviour during normal operation as well as during incidences, and
3. the lack of inspection, maintenance, and repair crews, as they are known in plants or in the aerospace domain.

It is expected that HRA will become more and more relevant for the functional safety of automotive systems, too, specifically because of the increase of high-voltage systems in electric and hybrid cars, and because of the increasing integration of active safety systems and driver assistance systems.

Moreover, there are special approaches to dynamic ZSA using MoCaS in the automotive domain, too. For example, [24] considers the influence of dynamic system behaviour on the

system's failure behaviour by taking time-dependent failure data into account. As these approaches require comparably high effort, they are used only for special cases and are not (yet) widespread.

2.3.3 Dynamic FTA – Other Approaches

From here on this thesis on dynamic FTA focusses on “dynamic” in the sense of representation of event sequences.

Known approaches to extending the FTA by dynamic effects typically are either simulations, or they automatically transform the fault tree model into a markov model, and then solve the resulting differential equation system.

The well known DFT approach [37] is based on modularizing the fault tree into static and dynamic modules, that are then calculated using *binary decision diagrams* (BDD) [47, 48] and markov chains. Static modules consist only of Boolean fault tree gates and events; dynamic modules also include dynamic fault tree gates. The latter are used to represent effects like sequences, or cold, warm, and hot redundancies, or trigger events. Figure 2.2 shown the main steps of this approach and compares them to the conventional static FTA.

The DFT method is included into numerous fault tree tools in differing completeness; e.g. in DIFTree [49] or Galileo [38], as well as in several commercial FTA tools like Isograph Faulttree+ [50], ITEM Toolkit [51], or RELAX Reliability Studio [52]. DFT are also mentioned in the recent edition of the *Fault Tree Handbook* [31].

A similar approach is presented in [53], which uses *dynamic bayesian networks* instead of creating and solving markov chains, a method for reducing calculatory costs.

Another alternative in [54] solves DFT modules with modified BDD, which are called *zero-suppressed binary decision diagrams*; this approach requires to manually include sequence infor-

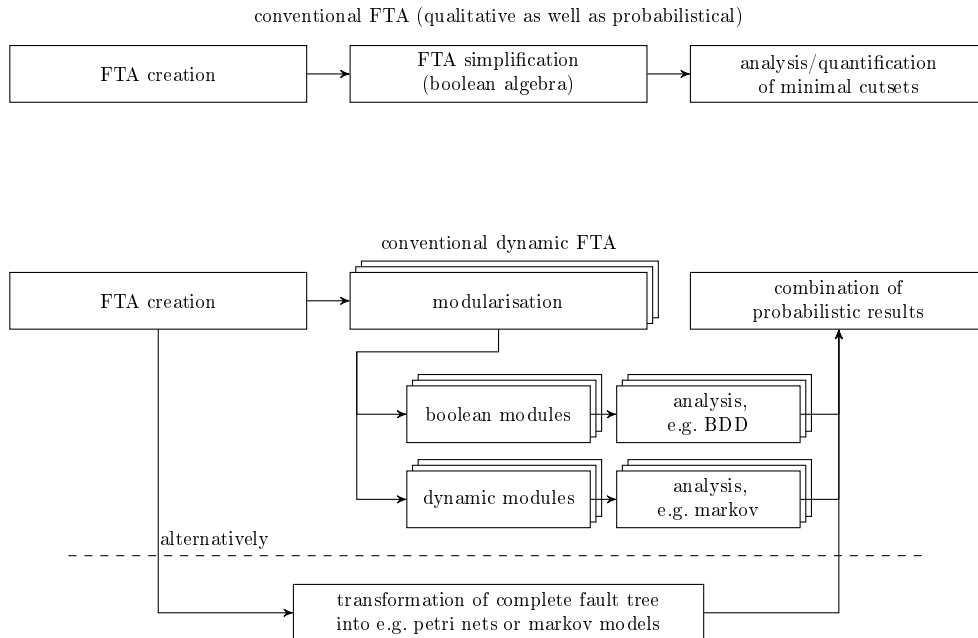


Figure 2.2: Main steps in conventional FTA (top) and - for comparison - main steps of a conventional dynamic fault tree extension using state-based modelling.

mation into the relevant minimal cutsets, instead of using markov models. This manual step limits the use of dynamic gates to relatively simple structures, though. Another similar such method is discussed in [55].

The approach introduced in [56, 57] is based on *Boolean logic driven markov processes* (BDMP) and, compared to the listed approaches from above, improves qualitative system analysis, and to some extent also allows taking repairable components into account.

A different approach to dynamic FTA based on petri nets, and without markov models, is chosen in [58] and [59, 60]; a further possibility are *state-event-fault-trees* given in [61].

Discussion

All these approaches to dynamic FTA are based on transforming the original fault tree model into state-based models. The latter are able to consider temporal dependencies and thus event sequences, too. The different approaches differ in their choice of transformation method – on the one hand, the complete fault tree is transformed; on the other hand, modularization and transformation only of those sub-trees that carry relevant dynamic data –, and they differ in their choice of state-based method.

But they have in common that, firstly, their calculatory cost grows exponentially with the size of their dynamic modules. Newer methods in [62, 63] reduce the time needed for the actual modularization, so that the calculatory effort grows only linearly with the number of modelled elements. But the complexity for solving the markov chains is always $O\{K \cdot N^3\}$ [64]. K is dependent on the number of computation-steps, and thus from the mission time and the calculations precision. And N is dependent on the number of states within the markov model; this number is in the range of $N = n^n$ for n elements under consideration. This *state explosion* [65] requires modularization with as small dynamic modules as possible. On the other hand, these markov models and their resulting differential equation systems can, in many cases, only be solved approximately, even despite of modularization (see e.g. [64]).

Secondly, modularization requires that the modules are independent from each other. This limits the dynamic dependencies between the system's elements that can be considered in the model; or it implies increasing the size of the dynamic modules – with the described negative impact on calculatory effort.

Thirdly, qualitative analyses are not possible, or possible only for very simple structures. This is owed to the transformation into the state space which does not follow the real system architecture as closely as the Boolean system model. One of the main benefits of the FTA is therefore missing in state based models: they can not “automatically” transform the modelled structure into a minimal form. For example, the DFT provides – depending on its specific implementation – either “normal” Boolean minimal cutsets without any event sequence information, or provides minimal cutsets with “meta-events”, that cover complete markov models without further breaking them apart.

Fourthly, state based models lack the “user-friendliness” of Boolean methods, also resulting from the Boolean model's closeness to the real system architecture. Instead, components and their dependencies are, for example, expressed by states and state-transitions (in markov models), or by places and transitions and marks (in petri nets). Figure 2.3 shows an example. One effect resulting from these differences is that state-based methods and models are less easy readable, less comprehensible, less easy in maintenance, and less scalable than the conventional FTA [67].

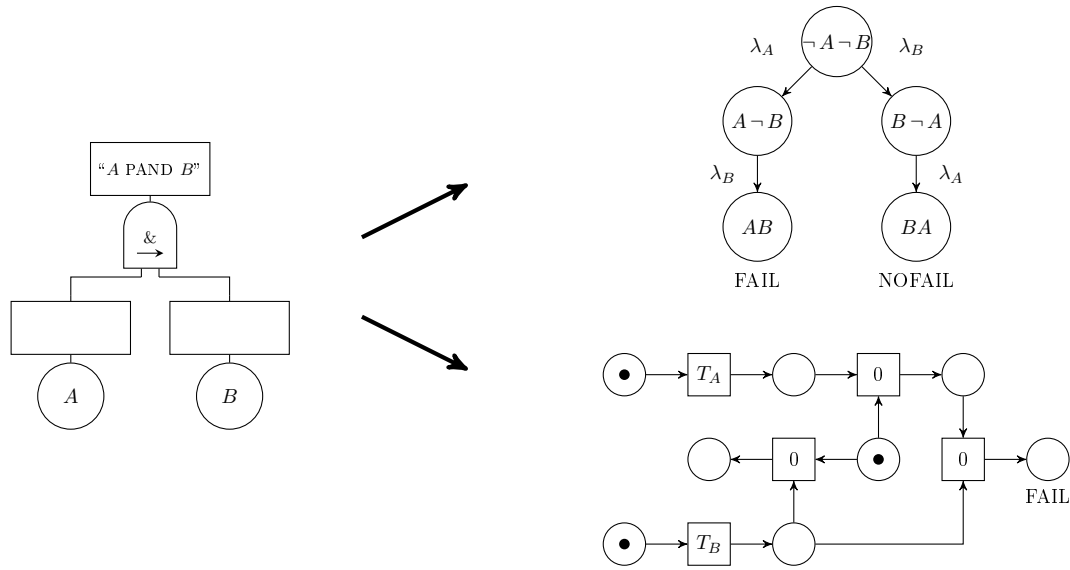


Figure 2.3: On the left side, a PAND gate with inputs A and B , that need to occur in the sequence “ A first, and then B ”, in order for the gate event to occur. The symbol used is not the one used in [5] but is taken from the TFTA approach in chapter 4. The top right side shows a markov model from [66], which is equivalent to the PAND gate; the bottom right side shows a petri net from [59], which is also equivalent to the PAND gate (T_A and T_B represent the time-to-failure of A and B).

2.3.4 Dynamic FTA – Based on a Temporal Failure Logic

Another possibility to include temporal dependencies is to use a *temporal logic* that extends the conventional Boolean logic. A temporal logic describes not only structural combinations of different events – that is the Boolean approach –, but also has a concept of time. The latter is used to make statements on the points in time at which events occur, and to include such statements into the logic function.

Applied to the field of reliability and safety, there are several approaches to use temporal logic for fault trees. One early approach of describing event sequences is found in [68]. It concentrates on probabilistic modelling aspects for individual event sequences; this is an approach that has later been revived and refined, e.g. in [59] and [69]. All these works do not expand onto a general temporal logic, which goes beyond taking individual event sequences into account. Therefore, they require that the relevant minimal failure sequences, that lead to the TOP event, have been found with other methods. This, of course, severely limits their application for complex projects.

The first version of the *fault tree handbook* [5] was a de facto standard for fault tree analysis for a long time; it also describes a so-called *priority AND* (PAND) gate. This gate is used exclusively for qualitative modelling of event sequences; probabilistically it is treated as a conventional AND gate. This approach again focusses on individual event sequences, and it does not provide a further and generic temporal logic. For example, it is not discussed, whether – and how – the fault tree structure shown on the left side of figure 2.4 may be simplified, and/or if it is equivalent to the structure shown on the right side of figure 2.4. In the Boolean model with AND instead of PAND gates, both fault trees are equivalent, as the Boolean distributive law – see (4.32) on page 31 – yields $(A \wedge B) \vee (A \wedge C) = A \wedge (B \vee C)$

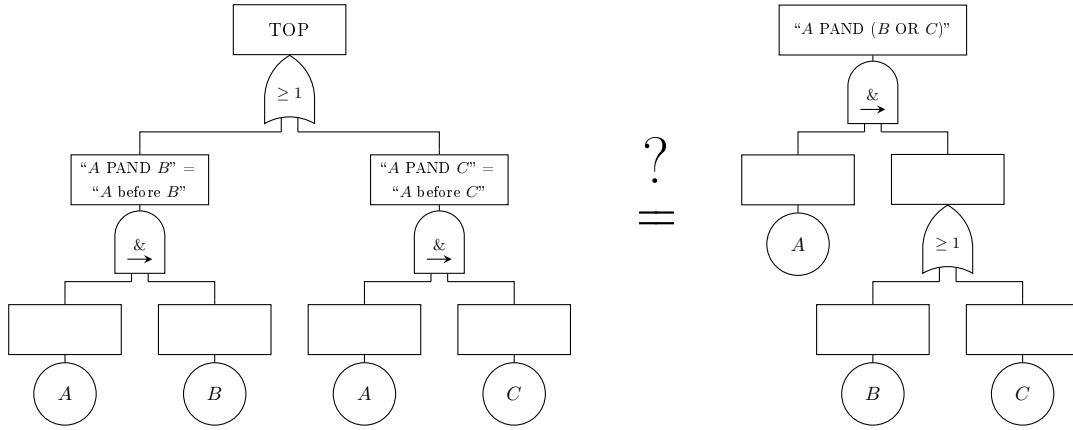


Figure 2.4: Questions on the state of the art of dynamic FTA using temporal logic. As the PAND gates introduced in [5] lack a universal temporal logic, it is undefined, whether both shown fault trees are equivalent or not. The PAND gates' symbols used in this figure are not the ones from [5], but from the TFTA approach presented in chapter 4.

The interval-based temporal logic of the so-called *AND-Then gates* in [70] pursue a broader approach, as well as the work presented in [71, 72] and the so-called *temporal fault trees* in [73]. They all stem from the field of formal fault tree analysis, which is mainly motivated by adopting the conventional fault tree analysis method, so as to model software based systems and their “failures”. Failure analysis of software based systems is fundamentally different from the conventional and hardware orientated ZSA, especially because of their very different failure mechanisms. An overview on the state of the art of FTA for software based systems is given in [74]. Because of the high dynamic of software based systems, the temporal logics presented in the works above are also complex and complicated; furthermore, their application is quite different to conventional FTA, because of their very strict definitions.

In earlier work, Heidtmann interpreted *modal logic* [75], which originates in the field of theoretical philosophy, for reliability modelling, see [11] and [34]. His temporal logic describes event sequences not directly, but asserts so-called *anytime-* and *always-relationships* between events. Using these, many temporal dependencies and contexts may be portrayed, including event sequences. Heidtmann discusses the qualitative as well as the probabilistic application of his temporal logic, and he is not limited to the fault tree method. On the other hand and because of its power, his logic involves comparably complex models and calculations.

The dedicated aim of the *Pandora* approach in [76, 77] is to provide a “useable” method that is similar to conventional FTA. The term “Pandora” puns on the figure from greek legend, as well as it is a composite of “Priority AND” and the greek term $\acute{\omega}\rho\alpha$ (*ora*), which means “time” [76]. Creation and analysis of Pandora fault trees is similar to conventional Boolean FTA. By using additional temporal gates – which are called PAND, SAND, and POR –, a temporal failure function of the TOP event is built. This function is then transformed into a minimal form by applying temporal logic simplification laws that are sketched in [77]. Central to these laws is the concept of so-called “doublets”. A doublet describes the temporal relationship between exactly two events, and is itself treated like a basic event. Temporal relationships are given only relatively to each other, i.e. the absolute points in time when events occur are not considered.

The minimal form is the equivalent to the minimal cutsets in conventional FTA; it allows a qualitative analysis of the failure behaviour including event sequence information. The concept of doublets simplifies the analysis greatly; but it also limits the Pandora approach in terms of probabilistic analysis, specifically because it leaves unresolved (temporal) dependencies between doublets. For example, in Pandora [77] the expression “ A occurs first, and then B and C occur” is written as

$$A \vec{\wedge} (B \wedge C) = \left[(A \vec{\wedge} C) \wedge (B \vec{\wedge} C) \right] \vee \left[(A \vec{\wedge} B) \wedge (B \vec{\wedge} C) \right] \vee \left[(A \vec{\wedge} B) \wedge (C \vec{\wedge} B) \right]. \quad (2.1)$$

Instead of the original Pandora notation, the notation from chapter 4 is used here, in order to improve comparability of the results. Each term in round brackets on the right side indicates one doublet.

These doublets allow qualitative analyse, but they can not be simply quantified, as shown by the following considerations.

A Boolean conjunction, e.g. $(A \wedge C) \wedge (B \wedge C)$, must not, in general, be quantified by simple multiplication of the individual event probabilities; i.e.

$$F_{(A \wedge C) \wedge (B \wedge C)} \neq (F_A \cdot F_C) \cdot (F_B \cdot F_C), \quad (2.2)$$

if it is not given in a minimal form, already, or the individual events are not independent from each other. If these conditions are satisfied, e.g. after transforming into

$$(A \wedge C) \wedge (B \wedge C) = A \wedge B \wedge C, \quad (2.3)$$

then a direct quantification is possible.

$$F_{(A \wedge C) \wedge (B \wedge C)} = F_{A \wedge B \wedge C} = F_A \cdot F_B \cdot F_C. \quad (2.4)$$

In analogy, Pandora expressions, like the one shown above, must not be quantified directly. For example, the “joint” event C in both doublets, i.e. an unresolved dependency between both doublets, is the reason for

$$F_{(A \vec{\wedge} C) \wedge (B \vec{\wedge} C)} \neq F_{(A \vec{\wedge} C)} \cdot F_{(B \vec{\wedge} C)}. \quad (2.5)$$

The TFTA approach presented in this work adopts some aspects of Pandora. But the TFTA goes beyond Pandora by (among others)

- providing a complete and systematic set of logic transformation laws of universal validity and applicability, where Pandora only sketches temporal logic rules in [77], and
- allowing probabilistic, as well as qualitative modelling and analysis, where Pandora stays qualitative, and
- not pursuing the concept of doublets, that is not well-suited for probabilistic analysis, and
- not using a POR operator.

The differences from that may be demonstrated by comparing the Pandora expression from above with an equivalent expression according to the TFTA approach. Anticipating the chapters below, the latter is given as

$$\begin{aligned} A \vec{\wedge} (B \wedge C) = & \left[A \vec{\wedge} B \vec{\wedge} C \right] \vee \left[B \vec{\wedge} A \vec{\wedge} C \right] \vee \left[A \vec{\wedge} C \vec{\wedge} B \right] \vee \left[C \vec{\wedge} A \vec{\wedge} B \right] \vee \\ & \vee \left[(A \vec{\wedge} B) \vec{\wedge} C \right] \vee \left[A \vec{\wedge} (B \vec{\wedge} C) \right] \vee \left[(A \vec{\wedge} C) \vec{\wedge} B \right]. \end{aligned} \quad (2.6)$$

As shown in this thesis, these terms may be quantified directly – and they may also be transformed into a more compact form in order to reduce the calculatory effort:

$$A \vec{\wedge} (B \wedge C) = \left[(A \wedge B) \vec{\wedge} C \right] \vee \left[(A \wedge C) \vec{\wedge} B \right] \vee \left[A \vec{\wedge} (B \bar{\wedge} C) \right]. \quad (2.7)$$

The right side expressions are mutually exclusive (disjoint), thus

$$\begin{aligned} F_{A \vec{\wedge} (B \wedge C)}(t) &= F_{(A \wedge B) \vec{\wedge} C}(t) + F_{(A \wedge C) \vec{\wedge} B}(t) + F_{A \vec{\wedge} (B \bar{\wedge} C)}(t) = \\ &= \int_0^t \left(F_A(\tau) F_B(\tau) f_C(\tau) + F_A(\tau) F_C(\tau) f_B(\tau) \right) \cdot d\tau. \end{aligned} \quad (2.8)$$

2.4 Summary

Conventional Boolean FTA is state of the art for systematic, top-down, and qualitative as well as probabilistic analysis of the failure behaviour of complex systems in several industries and application fields (see chapters 2.1 and 2.2).

The call for an improved consideration of time-dependencies lead to development of several extensions of the Boolean FTA in order to take into account dynamic effects and specifically *sequence dependencies*, see chapter 2.3.1. There are two main strategies for such consideration of event sequences: On the one hand the Boolean fault tree model is transformed into a state-based model, which allows the calculation of dynamic effects (see chapter 2.3.3). On the other hand, an extended and temporal logic is used instead of the Boolean (failure) logic, see chapter 2.3.4.

In the past several proposals for each of the two strategies were presented. Moreover, some of the state-based extensions are being used for solving real-world problems today. But by switching into the state-space these approaches loose some of the main advantages of conventional FTA, specifically with respect to the necessary calculatory effort, its intuitive useability, and its ability to provide meaningful qualitative analyses.

Very powerful but also very complex methods dominate the field of extensions by temporal logic; they stem mainly from research on applying the FTA on software. Further research is needed for improved useability, in order to convey the conventional Boolean FTA's “user-friendliness” onto dynamic FTA.

Figure 2.5 shows how the TFTA approach presented in this thesis fits into the state of the art, and it differentiates the TFTA from other methods.

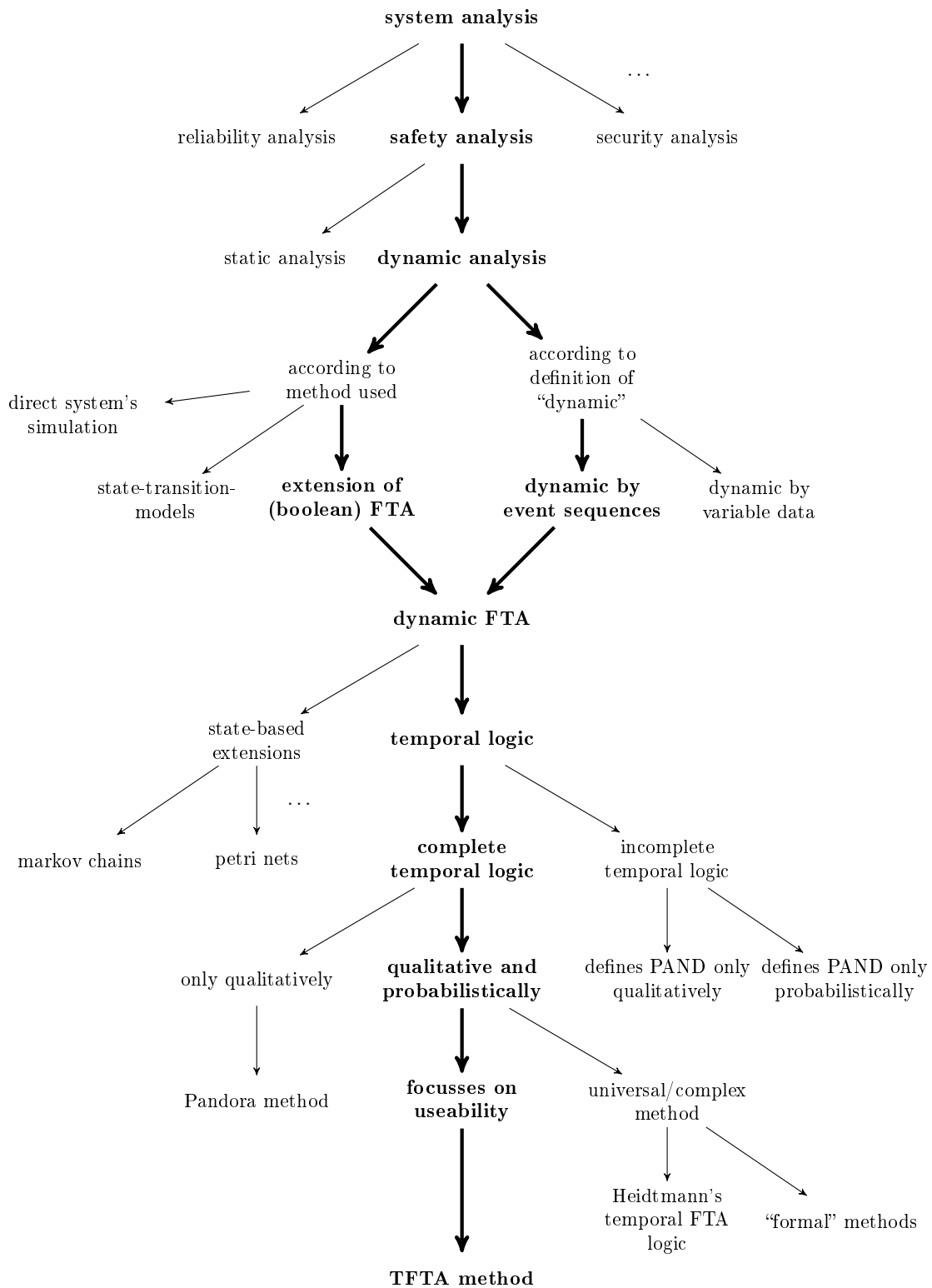


Figure 2.5: TFTA approach in comparison to the state of the art and other methods of considering “dynamic effects” within ZSA.

3 Problem Definition: Event Sequences in FTA without Modularization

Simplicity is the final achievement.

(Frédéric Chopin)

3.1 Demand for Improved Methods

3.1.1 Demand for Dynamic FTA

One of the FTA's vital objectives is the probabilistic evidence that the failure rate and failure probability of a system are lower than given target values. Practical experience shows that in many cases reaching these target values – derived from e.g. safety standards like IEC 61508 or ISO 26262 – is a close call. Modelling the same system with a dynamic FTA provides less conservative results than the conventional FTA; this, of course, helps to comply with probabilistic target values. It is much more credible to improve one's system analysis by using such a dynamic and more detailed method than to reach compliance with one's objectives by improving (reducing) the failure data input to the basic events; the latter is often hardly justifiable.

For systems with higher safety levels the conventional qualitative single failure analysis using FMEA is not sufficient [9, 78]. In such cases and for complex system architectures the qualitative FTA improves systematic understanding of multiple failure interaction. For example, it is very efficient to improve the safety of programmable systems by making the conditions of switching elements depend on sequential information. Fail-activation is reduced as only certain sequences of trigger events are relevant. In many cases such sequential conditions can be added into integrated circuits with only negligible costs. When compared to the conventional FTA, an FTA that takes such sequences into account can then provide a much more meaningful view on the system under consideration.

Chapter 6 shows an example system where conventional Boolean fault tree modelling and analysis provides only unprecise results.

3.1.2 Demand for Improved Dynamic FTA

Dynamic extensions to FTA, as listed in chapter 2.3, aim at the correct probabilistic calculation of fault trees; this is especially true for the state based methods like DFT. Chapter 6.3 shows an example where the DFT succeeds in this respect and thus proves to be a real improvement when compared to the conventional Boolean FTA.

Criticism of state based extensions comprises mainly from the following aspects:

- state based extensions are limited in their use for qualitative analysis of sequence effects. This comes from the forced change between methods with Boolean fault tree logic on the one hand and a state based dynamic model on the other hand.
- they are limited in case of interdependencies between dynamic and non-dynamic parts (modules) of the same fault tree.
- probabilistic calculation is rather costly and approximations are not easy to identify and use.

Practical experience shows that there is a certain correlation between the necessities of probabilistic and qualitative analyses of dynamic effects. Therefore, from an effort point of view it is beneficial to cover both aspects with the same modelling method. Methods are needed that allow both analyses with reasonable effort and ideally also allow a step wise workflow: first the results are only approximated, then the most important contributors are identified, and then only for those the more complex but exact calculations are done.

3.1.3 Remarks on Using Dynamic FTA

In general, an analysis' effort and its benefit must not be disproportionate to each other even if there is a very understandable quest to model the reality (which is dynamic, see chapter 2.1.2.1) as exact and detailed as possible. Today there are several attempts to extend the Boolean FTA with dynamic effects and event sequences; but many of those extensions are limited to simple and mostly academic examples. This is especially true for approaches based on a temporal logic; their very high complexity conflicts with their practical useability.

Useability, (relative) ease of use, and scalability are three critical success factors of the conventional FTA; and they have added tremendously to the FTA being first choice for safety and reliability analyses in many domains.

In order to transfer this success, the dynamic FTA needs to satisfy the following generic requirements:

- real system effects must translate into the model's logic easily,
- the actual implementation into a fault tree needs to be possible with reasonable effort,
- qualitative as well as probabilistic calculations must be possible without changing the analysis method,
- computing time must be reasonable,
- the fault tree as well as its results must be easily readable and comprehensible,
- scalability and possibility to detail and extend parts of the fault tree.

3.2 Concept

3.2.1 Requirements for TFTA

By taking useability and practical considerations into account the following is required from the new TFTA method:

1. The temporal TFTA logic shall be able to model sequence dependencies between events.

2. The temporal TFTA logic shall be a detailing (extension) of the Boolean logic.
3. The TFTA shall be similar to the conventional FTA regarding notation, abstract concept, workflow, work products.
4. The qualitative TFTA shall provide minimal event sequences similar to the Boolean minimal cutsets. Each “minimal cutset sequence” shall consist of “temporal conjunction terms” similar to the Boolean AND term but including event sequence information. The TOP or system failure function shall then consist of such “minimal cutset sequences” given in “temporal disjunctive normal form”.
5. In order to allow for probabilistic analysis the “minimal cutset sequences” shall be disjoint (i.e. mutually exclusive); this allows for easy quantification by convolution of the failure densities/frequencies.
6. In order to reduce calculation efforts the TFTA shall support step-wise modelling: a first step provides only approximations; more exact calculations follow only for the most important contributors. It shall be possible to calculate exact results if necessary.

Assumptions on TFTA

The following discussions are based on two assumptions:

1. fault trees are monotone (sometime also called coherent) and
2. all component failures are non repairable.

3.2.2 TFTA – Step by Step

Figure 3.1 shows the TFTA workflow with its multiple steps. First, there is the two step qualitative transformation of the initial logic expression into a minimal and later disjunct form; in a second step, this is then quantified probabilistically. This workflow is very similar to the workflow of conventional FTA; there, too, minimal cutsets need not automatically be mutually exclusive. The TFTA workflow is split into two steps because of the potentially very high effort necessary for transforming a minimal temporal expression into mutually exclusive terms.

The structure of chapter 4 is influenced by this workflow steps, too; chapter 4.1 provides the notation of the temporal logic; chapter 4.2 provides the TFTA’s (temporal) rules of transformation; chapter 4.3 describes the transformation into mutually exclusive sequences; and chapter 5 provides the probabilistic evaluation of temporal expressions.

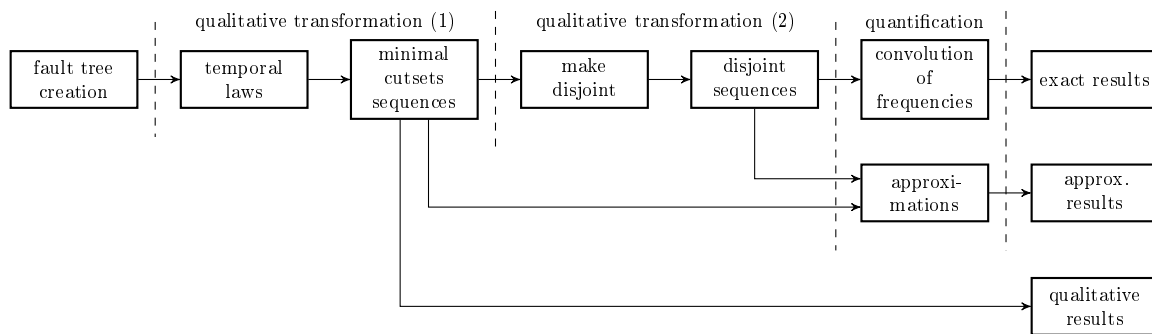


Figure 3.1: Step by step workflow of the TFTA with its two-step transformation of a temporal expression into a minimal and then mutually exclusive form, and probabilistic quantification. Approximations are possible, first, based on the mutually exclusive event sequences or, second and a little more unprecise, directly from the minimal event sequences.

4 Temporal Fault Tree Analysis (TFTA): A New Approach to Dynamic FTA

Time is the worst place, so to speak,
to get lost in.

(Douglas Adams)

This chapter describes the *temporal fault tree analysis* (TFTA) which extends the Boolean FTA and allows analysis of event sequences.

- Chapter 4.1 presents the notation of the new temporal TFTA logic. Specifically, there are two new temporal operators corresponding to two temporal fault tree gates.
- At the heart of the new temporal logic there are several rules of transformation (“temporal logic laws”) described in chapter 4.2. They allow the transformation of a temporal expression into its temporal disjunctive normal form (TDNF).
- Chapter 4.3 discusses minimal and disjoint temporal expressions.
- There is an extended form of temporal expressions, as shown in chapter 4.4, which reduces the effort necessary for describing and calculating complex temporal failure functions – especially if such failure functions only include few real temporal relationships between events.

4.1 TFTA Notation

First of all, some remarks on the terms used: In the fault tree method basic events represent atomic failure events of real life entities (i.e. systems, components, parts, functions). Likewise, fault tree gates represent non-atomic “higher level” failure events. The terminology is sometimes confused so that there is no discrimination between “incidence of a real world failure event” and “fault tree event becomes *True*”, where the latter represents the real life event in the fault tree model.

4.1.1 Boolean Algebra and the FTA Failure Logic

In the context of FTA events are failure events. Contrary to uses of the Boolean algebra for reliability calculations, the FTA therefore uses a *negated logic* [14, chapter 14.4.2]. In

the following text negating all events in their written form is omitted for reasons of better readability. For all failure events

$$X_i = \begin{cases} \text{True or 1} & \text{entity } i \text{ has failed} \\ \text{False or 0} & \text{entity } i \text{ is operational} \end{cases} . \quad (4.1)$$

For the TFTA approach most of the Boolean logic and its application on the fault tree stays the same:

The *conjunction* using the *AND operator* and

$$X_{\text{AND}} = A \wedge B \quad (4.2)$$

is *True*, if and only if both events *A* and *B* are *True*. In fault trees the conjunction is represented by AND gates.

The *disjunction* using the *OR operator* and

$$X_{\text{OR}} = A \vee B \quad (4.3)$$

is *True*, if either only event *A* or only event *B* is *True*, or if both events are *True*. In fault trees the disjunction is represented by OR gates.

The *negation* using the *NOT operator* and

$$X_{\text{NOT}} = \neg A \quad (4.4)$$

is *True*, if and only if event *A* is *False*. The shorter $A \neg B$ is used below instead of $A \wedge \neg B$. In fault trees the negation is represented by NOT gates.

4.1.2 Temporal Logic Operators

The TFTA uses two temporal operators and their corresponding gates in addition to the Boolean operators and gates in order to describe temporal event relationships (see figure 4.1).

PAND: The Sequence of Events

The *PAND operation* (*Priority AND*) using the *PAND operator* and

$$X_{\text{PAND}} = A \vec{\wedge} B \quad (4.5)$$

is *True*, if and only if

- both events *A* and *B* are *True* and
- *A* has become *True* before *B* has become *True*.

Therefore, PAND describes a chronology of events becomming *True* after each other. In fault trees the PAND operation is represented by PAND gates.

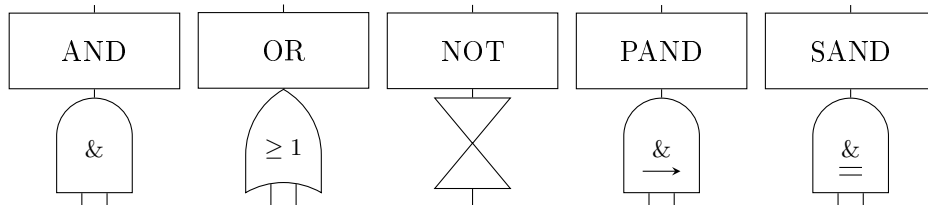


Figure 4.1: Fault tree gates of the TFTA: Boolean gates (left) and temporal gates (right)

SAND: Concurrency of Events

The *SAND operation* (*Simultaneous AND*) using the *SAND operator* and

$$X_{\text{SAND}} = A \bar{\vec{\wedge}} B \quad (4.6)$$

is *True*, if and only if

- both events A and B are *True* and
- A and B have become *True* simultaneously.

Therefore, SAND describes events becoming *True* exactly at the same time. In fault trees the SAND operation is represented by SAND gates.

Remark: PAND as well as SAND uses time indications relatively, i.e. no statement is made on the absolut (real) time at which an event becomes *True*.

4.1.3 Boolean and Temporal Operations Visualized as Sets

Figure 4.2 shows the different operators as sets and illustrates the relationships among them. First, there are two event A and B symbolized as sets. If A and B are the operands to AND and OR operators (i.e. they are inputs to Boolean AND and OR gates in a fault tree), then two sets result: $A \wedge B = B \wedge A$ (intersection) and $A \vee B = B \vee A$ (union). If A and B are the operands to PAND and SAND operators (i.e. they are inputs to temporal PAND and SAND gates in a temporal fault tree), then three sets result: $A \vec{\wedge} B$ and $B \vec{\wedge} A$ and $A \bar{\vec{\wedge}} B = B \bar{\vec{\wedge}} A$. Note, that negated events and their corresponding “sets” are not shown here. The depiction in figure 4.2 allows a first qualitative statement on the meaning of temporal operators/gates.

According to (4.5) and (4.6) PAND and SAND events are real subsets of the Boolean conjunction $A \wedge B = B \wedge A$ (“...both events A and B are *True* ...”). There are three possible sequences how two events A and B can “both be *True*” (see the law of completion in chapter 4.2).

As sets this may be written as

$$A \vec{\wedge} B \subset A \wedge B, \quad A \bar{\vec{\wedge}} B \subset A \wedge B, \quad B \vec{\wedge} A \subset A \wedge B, \quad (4.7)$$

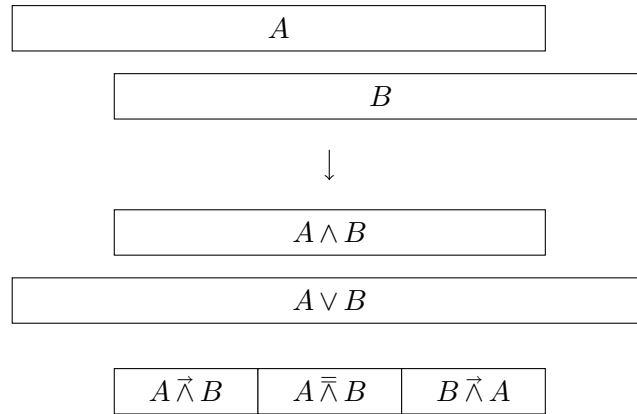


Figure 4.2: Temporal operations from top to bottom: events A and B ; their intersection (AND) and union (OR); the three subsets defined by distinction between the possible event sequences (PAND and SAND).

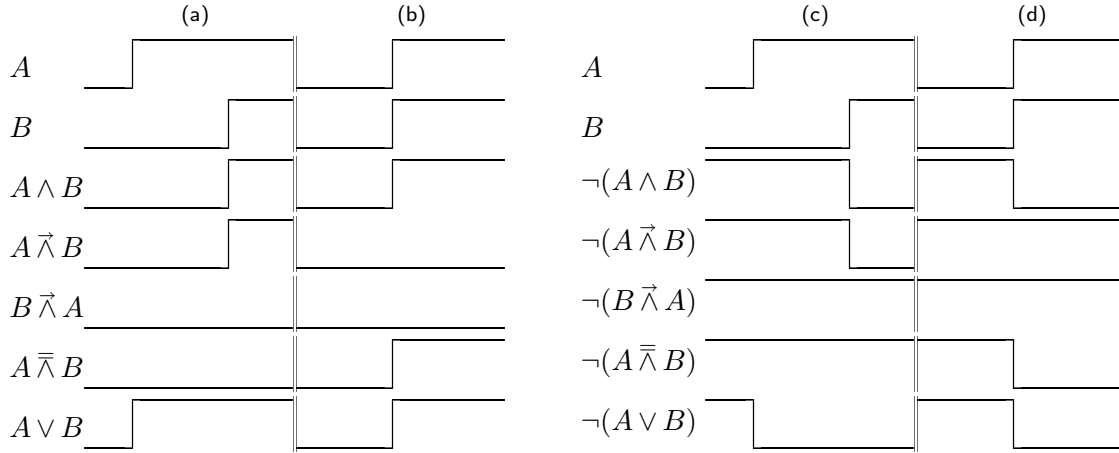


Figure 4.3: Temporal sequence of two events: In (a) and (c) event A becomes *True* before B (upper two rows); the following rows show which events formed by A and B become *True* at which time. In (b) and (d) events A and B become *True* simultaneously.

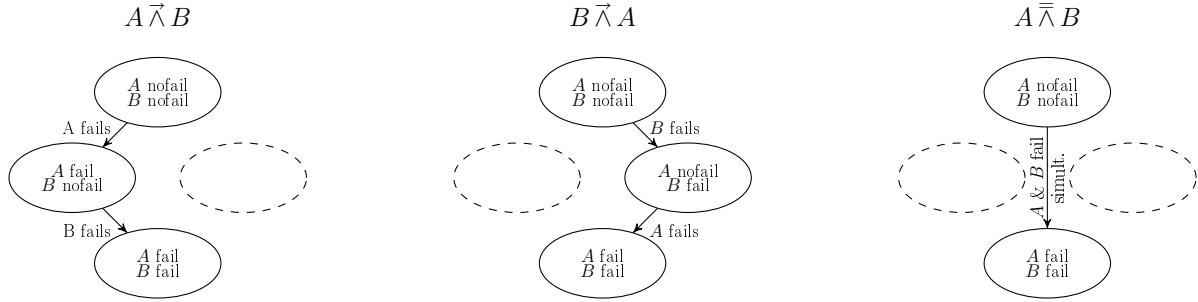


Figure 4.4: Illustration of the three possible sequences of state transitions (which are mutually exclusive) that lead to failure of both components of the example system in figure 2.1.

$$A \wedge B \subset A, \quad A \wedge B \subset B. \quad (4.8)$$

Events $A \overset{\rightarrow}{\wedge} B$, $B \overset{\rightarrow}{\wedge} A$, and $A \overset{\leftarrow}{\wedge} B$ are pairwise mutually exclusive, i.e. there is no intersection between them (see chapter 4.3):

$$A \overset{\rightarrow}{\wedge} B \perp A \overset{\leftarrow}{\wedge} B, \quad A \overset{\rightarrow}{\wedge} B \perp B \overset{\rightarrow}{\wedge} A, \quad A \overset{\leftarrow}{\wedge} B \perp B \overset{\rightarrow}{\wedge} A. \quad (4.9)$$

4.1.4 Temporal Operations: Timing Behaviour

Temporal sequence diagrams illustrate (temporal) relationships between events. Figure 4.3 shows logic levels over time for Boolean and temporal operators used in the TFTA. In general, events may become *True* in sequence or simultaneously (see sub-figures (a) and (c) and (b) and (d) respectively).

The possible failure sequences in a system which result from those timings may be shown e.g. with state diagrams. In a simple example system consisting of two redundant components (see state diagram in figure 2.1), where events may become *True* after each other or simultaneously, there are the three possible state transition sequences which were already mentioned and which

are shown in figure 4.4. These sequences correspond to the two PAND operations $A \vec{\wedge} B$ and $B \vec{\wedge} A$ on the one hand and the SAND operation $A \overline{\wedge} B$ on the other hand.

From page 33 on further examples of temporal sequence diagrams are compared with other methods of illustration.

4.1.5 Syntax of Temporal Expressions

A logic expression with at least one temporal operator is called *temporal-logic expression* or shorter: *temporal expression*.

In conventional FTA a Boolean expression which is represented by the fault tree's TOP event is called *Boolean failure function* and is symbolized by φ . In the TFTA the TOP event represents a temporal expression which is called *temporal failure function* and is symbolized by its own symbol ϖ for better discrimination in the following text.

The next sections explain elements of a temporal logic grammar as used by TFTA. This grammar is summarized in table 4.1. The temporal logic's operators $\{\wedge, \vee, \vec{\wedge}, \overline{\wedge}, \neg\}$ are used as terminal symbols.

Atomic Events/Basic Events

Atomic events are the smallest event entities in temporal expressions, and are not further dividable. Within the temporal fault tree they are represented by basic events which do not differ from those basic events used in conventional FTA. Particularly, probabilistic (failure) data like failure rates may be assigned to them.

The formal grammar of the temporal logic uses the **ae** token for atomic events.

Negated atomic events with token **nae** are – as the name suggests – the negation of atomic events:

$$\text{nae} \rightarrow \neg \text{ae} \quad . \quad (4.10)$$

Within the TFTA negated events have a special meaning, see chapter 4.2.8.

General Temporal Expressions

In general, a temporal expression either consists of a basic event, or consists of two other temporal expressions, which are connected by a temporal (including Boolean) operator, or consist of a negation of another temporal expression. Therefore

$$\begin{array}{lcl} \text{tt} & \rightarrow & \text{ae} \\ & & \text{tt} \wedge \text{tt} \\ & & \text{tt} \vee \text{tt} \\ & & \text{tt} \vec{\wedge} \text{tt} \\ & & \text{tt} \overline{\wedge} \text{tt} \\ & & \neg \text{tt} \end{array} \quad | \quad . \quad (4.11)$$

Aside from the additional temporal operators this corresponds to the formal representation of Boolean expressions.

This general form is not suited for direct qualitative or probabilistic analysis. From chapter 4.2 on transformation laws for temporal expressions are described that allow to transform any temporal expression into a TDNF – which in turn allow further analysis. The following sections explain the structure of this TDNF.

Token	Description	Format	Example
ae	atomic event (basic event)	-	X Y Z
nae	negated atomic event	$\neg ae$	$\neg X$
ce	core event	ae $ce \bar{\bar{ae}}$	see above $X \bar{\bar{Y}}$ $X \bar{\bar{Y}} \bar{\bar{Z}}$
nce	negated core event	nae $nce \wedge nae$	see above $\neg X \wedge \neg Y = \neg X \neg Y$ $\neg X \wedge \neg Y \wedge \neg Z = \neg X \neg Y \neg Z$
es	event sequence	ce $es \bar{\bar{ce}}$	see above $X \bar{\bar{Y}}$ $(X \bar{\bar{Y}}) \bar{\bar{Z}}$
nes	event sequence with negated events	$nce \wedge es$	$\neg X \wedge Y$ $\neg X \wedge (Y \bar{\bar{Z}})$ $(\neg X \neg Y) \wedge Z$ $(\neg X \neg Y) \wedge (A \bar{\bar{Z}})$
tdnf	temporal expression in TDNF	es nes $tdnf \vee tdnf$	see above see above $X \vee Y$ $[\neg X \wedge (Y \bar{\bar{Z}})] \vee [(X \bar{\bar{Y}}) \bar{\bar{Z}}]$
ece	extended core event	$ae \wedge ae$ $ece \wedge ae$	$X \wedge Y$ $X \wedge Y \wedge Z$
ees	extended event sequence	ece $ees \bar{\bar{ece}}$ $ees \bar{\bar{ce}}$ $es \bar{\bar{ece}}$	see above $(X \wedge Y) \bar{\bar{(A \wedge Z)}}$ $(X \wedge Y) \bar{\bar{(A \bar{\bar{Z}})}}$ $X \bar{\bar{(Y \bar{\bar{Z}}) \bar{\bar{(A \wedge B)}}$
nees	extended event sequence with negated events	$nce \wedge ees$	$\neg Z \wedge (X \wedge Y)$ $\neg Z \wedge [(X \wedge Y) \bar{\bar{(A \wedge B)}}$ $\neg Z \wedge [(X \wedge Y) \bar{\bar{(A \bar{\bar{B}})}}$ $(\neg X \neg Y) \wedge [X \bar{\bar{Y}} \bar{\bar{(A \wedge B)}}$
etdnf	temporal expression in extended TDNF	ees nees $etdnf \vee tdnf$ $etdnf \vee etdnf$	see above see above $(X \wedge Y) \vee Z$ $[\neg X \wedge (Y \wedge Z)] \vee [(X \bar{\bar{Y}}) \bar{\bar{Z}}]$ $(X \wedge Y) \vee (A \wedge B)$ $[\neg A \wedge (X \wedge Y)] \vee [Z \bar{\bar{(A \wedge B)}}$
tt	generic temporal expression	ae $tt \wedge tt$ $tt \vee tt$ $tt \bar{\bar{tt}}$ $tt \bar{\bar{tt}}$ $\neg tt$	see above $(A \vee B) \wedge (C \bar{\bar{D}})$ $A \vee \neg(C \bar{\bar{D}})$ $(A \vee B) \bar{\bar{(C \vee D)}}$ $(A \vee B) \bar{\bar{(C \vee D)}}$ $\neg(C \bar{\bar{D}})$

Table 4.1: The syntax of temporal expressions: the more complex tokens are based on the token of an atomic event (basic event) as an entity which is not further dividable; complex tokens are: core events, event sequences and temporal expressions in TDNF; they are composed in multiple ways. The examples given do not include all possible combinations. The lower part of the figure shows temporal expressions in a more generic form; those need to be transformed for further analysis.

4.1.5.1 Temporal Disjunctive Normal Form (TDNF, Sum of Products)

Core Events

In the temporal logic *core events* describe that one or more events become *True* at a certain point in time. Negated core events indicate that at a given time one or more events have not (yet) become *True*. Many equations in this thesis use K for core events.

A core event event is represented by token ce and consists of either one atomic event, or consists of a temporal expression (in braces), which itself consists of only SAND connected atomic events. More formally,

$$\begin{array}{lcl} ce & \rightarrow & ae \\ & & ce \bar{\wedge} ae \end{array} \quad | \quad (4.12)$$

A *negated core event* (token nce) consists of either one negated atomic event, or consist of a temporal expression (in braces), which itself consists of only AND connected negated atomic events. More formally,

$$\begin{array}{lcl} nce & \rightarrow & nae \\ & & nce \wedge nae \end{array} \quad | \quad (4.13)$$

Event Sequences

Event sequences are the temporal logic's equivalent of Boolean cutsets. They describe a temporal sequence of one or more core events. In analogy to the Boolean minimal cutsets, minimal event sequences (MCSS, see chapter 4.3.2) have a special significance in the temporal logic.

Event sequences with negated events are important for transforming temporal expressions into disjoint, i.e. mutually exclusive, terms. This is similar to the Boolean logic. Many equations in this thesis use ES for event sequences.

Event sequences are represented by the token es and either consist of exactly one core event, or consist of several PAND connected core events. More formally

$$\begin{array}{lcl} es & \rightarrow & ce \\ & & es \vec{\wedge} ce \end{array} \quad | \quad (4.14)$$

Additionally, there are event sequences with negated events consisting of exactly one negated core event, which is AND connected with exactly one event sequence. They are represented by the token nes . Therefore

$$nes \rightarrow nce \wedge es \quad . \quad (4.15)$$

Temporal Expressions in TDNF

Event sequences, connected by OR operators, provide the *temporal disjunctive normal form* (TDNF):

$$\varpi = \bigvee_{j=1}^{\zeta} ES_j = ES_1 \vee ES_2 \vee \dots \vee ES_{\zeta} . \quad (4.16)$$

The symbol ζ indicates the number of event sequences ES of ϖ , which themselves are not necessarily already in a minimal form. More formally,

$$\begin{array}{lcl} tdnf & \rightarrow & es \\ & & nes \\ & & tdnf \vee tdnf \end{array} \quad | \quad (4.17)$$

4.1.5.2 Extended TDNF (Sum of Products)

Temporal Expression in Extended TDNF

The extended TDNF of a temporal failure function ϖ is given as ζ *extended event sequences* eES_j which are connected by OR operators:

$$\varpi = \bigvee_{j=1}^{\zeta} eES_j = eES_1 \vee eES_2 \vee \dots \vee eES_{\zeta} . \quad (4.18)$$

This extended TDNF greatly simplifies the qualitative as well as probabilistic transformations and calculations. More formally,

$$\begin{array}{lll} \text{etdnf} & \rightarrow & \text{ees} \\ & & \text{nees} \\ & & \text{etdnf} \vee \text{tdnf} \\ & & \text{etdnf} \vee \text{etdnf} \end{array} \quad \begin{array}{l} | \\ | \\ | \\ . \end{array} \quad (4.19)$$

The extended TDNF consists of extended core events and extended event sequences with and without negated events.

Extended Core Events

An *extended core event* is represented by the token **ece** and consists of two or more AND connected atomic events. It is identical to the conventional conjunction of atomic events in Boolean algebra. Therefore,

$$\begin{array}{lll} \text{ece} & \rightarrow & \text{ae} \wedge \text{ae} \\ & & \text{ece} \wedge \text{ae} \end{array} \quad \begin{array}{l} | \\ . \end{array} \quad (4.20)$$

Extended Event Sequences

Extended event sequences with token **ees** either consist of exactly one extended core event or consist of only PAND connected extended core events or consist of a mixture of PAND connected normal and extended event sequences. Thus,

$$\begin{array}{lll} \text{ees} & \rightarrow & \text{ece} \\ & & \text{ees} \vec{\wedge} \text{ece} \\ & & \text{ees} \vec{\wedge} \text{ce} \\ & & \text{es} \vec{\wedge} \text{ece} \end{array} \quad \begin{array}{l} | \\ | \\ | \\ . \end{array} \quad (4.21)$$

Extended event sequences with negated events are defined as event sequences which consist of exactly one negated core event which is AND connected with exactly one extended event sequence; they are represented by the token **nees**. Formally,

$$\text{nees} \rightarrow \text{nce} \wedge \text{ees} . \quad (4.22)$$

The following chapters at first don't touch the subject of the extended form of temporal expressions. Chapter 4.4 then explains how the qualitative analysis is simplified by using extended event sequences. Chapter 5.4.2 discusses the probabilistic quantification of extended event sequences.

4.1.7.1 Normal Sequential Failure Trees (without SAND)

The sequential failure tree for a system comprised from n elements (e.g. components) has $n + 1$ levels with $\binom{n}{i} \cdot i!$ nodes on each level $i \in \{0, 1, \dots, n\}$, see figure 4.5. Each node represents one specific system state r and may be expressed as vector $\vec{K}_r = (X_1, X_2, \dots, X_n)$; all elements that are not failed in this system state are written with 0 (*False*), and all failed elements are written as $1, 2, \dots, i$ according to the failure sequence that lead to this system state.

For example, the sequence $A \vec{\wedge} B \vec{\wedge} C$, i.e. "A before B before C", corresponds to vector $\vec{K} = (1, 2, 3)$. the node on the top most level (level 0) has the zero vector $\vec{K} = (0, 0, \dots, 0)$.

A system's temporal failure function ϖ may be expressed as function of vectors \vec{K}_r :

$$\varpi(\vec{K}_r) = \begin{cases} 1 & , \text{ if the system is failed in state } r. \\ 0 & , \text{ if the system is not failed in state } r. \end{cases} \quad (4.25)$$

With the exception of the one node on level 0, every node \vec{K} has exactly one *predecessor node* \vec{K}' . With the exception of the nodes on the lowest level n , every node has at least one *successor node* \vec{K}'' .

Because of the definite sequence the following is always given:

$$\vec{K} > \vec{K}' . \quad (4.26)$$

According to this "vector inequation", no element in \vec{K} may be less than the corresponding element in \vec{K}' , and at least one element in \vec{K} must be greater than the corresponding element in \vec{K}' .

Accordingly,

$$\vec{K} < \vec{K}'' . \quad (4.27)$$

Taking the property of monotony into account, the following statement holds for failure functions:

$$\varpi(\vec{K}) \geq \varpi(\vec{K}') . \quad (4.28)$$

Furthermore, the property of monotony yields that if $\varpi(\vec{K}) = 0$ then the system function of a predecessor node \vec{K}' of node \vec{K} must also be $\varpi(\vec{K}') = 0$.

A node \vec{K} is a *minimal failure node* if the failure sequence that is represented by \vec{K} leads to a first-time failure of the system, i.e.

$$\varpi(\vec{K}) = 1 \quad \text{and} \quad \varpi(\vec{K}') = 0 . \quad (4.29)$$

The successor nodes of a minimal failure node are called *non-minimal failure nodes*. All successor nodes of a non-minimal failure node are also non-minimal failure nodes. And again, with the property of monotony the system function $\varpi(\vec{K}'')$ of all successor nodes of a minimal (or non-minimal) node $\varpi(\vec{K}) = 1$ must also be $\varpi(\vec{K}'') = 1$.

Sequential failure trees and the TFTA notation correspond to each other: *Nodes* (sequential failure tree) correspond to TFTA *failure sequences*; *minimal failure nodes* correspond to *MCSS*; *non-minimal failure nodes* correspond to *non-minimal failure sequences*.

Providing all minimal failure nodes (or, respectively, all MCSS) completely describes the TOP event of a temporal fault tree and its failure function ϖ .

The left side of figure 4.6 shows the simplified sequential failure tree (without SAND) of a system with three components A , B , and C and the failure function $\varpi = (C \bar{\wedge} B \bar{\wedge} A) \vee (B \bar{\wedge} C)$.

The sequential failure tree has $n + 1 = 4$ levels. Four of the $\sum_{i=0}^{i=n=3} \binom{n}{i} \cdot i! = 16$ possible nodes (without SAND) are minimal failure nodes which correspond to the four MCSS $A \bar{\wedge} B \bar{\wedge} C$ and $\neg A \wedge (B \bar{\wedge} C)$ and $B \bar{\wedge} A \bar{\wedge} C$ and $C \bar{\wedge} B \bar{\wedge} A$. In addition, there is a non-minimal failure node, corresponding to the failure sequence $B \bar{\wedge} C \bar{\wedge} A$.

Nodes that do not represent a system failure state are filled white, minimal failure nodes are filled black, and non-minimal failure nodes are crosshatched.

4.1.7.2 Sequential Failure Trees with Concurrent Events/SAND

The right side of figure 4.6 shows the sequential failure tree of a system with failure function $\varpi = (C \bar{\wedge} B \bar{\wedge} A) \vee (B \bar{\wedge} C)$; in this case SAND connections and corresponding nodes and transitions are also shown.

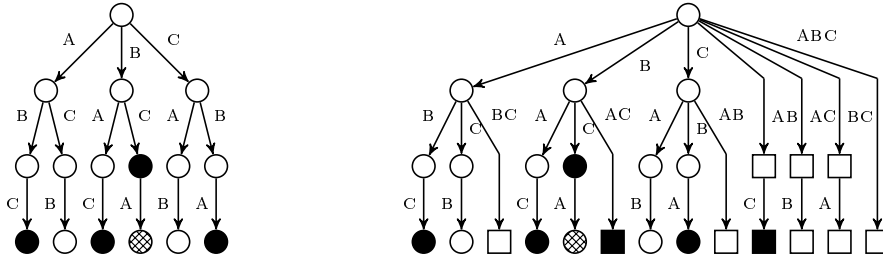


Figure 4.6: Sequential failure tree without SAND (left side) and with SAND (right side) of a system with failure function $\varpi = (C \bar{\wedge} B \bar{\wedge} A) \vee (B \bar{\wedge} C)$.

For better discrimination failure nodes (system failure states) without SAND connection are depicted as circles and failure nodes with at least one SAND connection are depicted as rectangles.

Besides that, the notation, as introduced in chapter 4.1.7.1, stays the same. For example, sequence $(A \bar{\wedge} B) \bar{\wedge} C$ corresponds to vector $\vec{K} = (1, 1, 2)$, and sequence $A \bar{\wedge} (B \bar{\wedge} C)$ corresponds to vector $\vec{K} = (1, 2, 2)$. Equations (4.25) to (4.29) also hold for sequential failure trees with SAND connections.

4.1.7.3 Using Sequential Failure Trees

Sequential failure trees allow an intuitive visualization of temporal expressions and thus ease their analysis:

- They directly illustrate temporal expressions, comparable to logic tables as illustrations of Boolean expressions. Moreover, different temporal expressions are equivalent, if they have identical sequential failure trees.
- They directly show if temporal expressions are minimal, or if they include each other, see chapter 4.3.2. Temporal expressions are minimal, if each of their sequential failure trees has at least one minimal failure node which is not a failure node in any of the other failure trees.

- They directly show if temporal expressions are mutually exclusive (disjoint), or if they have intersections, see chapter 4.3.3. Temporal expressions are mutually exclusive, if their failure trees have no failure node in common.

Two types of sequential failure trees are used below: the “explicit form” shown on the left side of figure 4.7, as well as a “compact form” shown on the right side of figure 4.7.

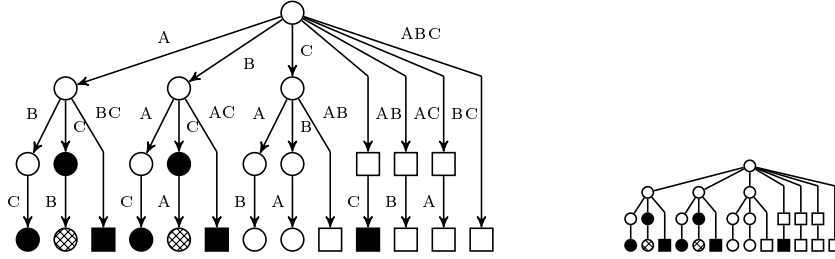


Figure 4.7: Explicit and compact forms of the same sequential failure tree with failure function $\varpi = (A \vec{\wedge} C) \vee (B \vec{\wedge} C)$. Both forms are used in this thesis.

Based on some examples, creating and using sequential failure trees is demonstrated from page 33 on; there, sequential failure trees are compared to other visualization methods, too. The appendix includes further explanations on sequential failure trees, see page 118.

Summary of Chapter 4.1:

The TFTA’s notation is based on the three Boolean operators AND, OR, and NOT, added by two new temporal operators PAND and SAND. Temporal expressions may be reduced to their sum of products form (OR connected event sequences), which is called TDNF and consists of PAND connected core events; the TDNF corresponds to the Boolean disjunctive normal form. The extended TDNF also allows AND connected core events, which reduces computing effort. Sequential failure trees allow the visualization of temporal expressions and show if temporal expressions are minimal or mutually exclusive (disjoint).

4.2 Laws of the TFTA Temporal Logic

The temporal logic rules of the TFTA method are an extension to conventional Boolean logic and algebra. These rules describe *temporal relationships* between events, i.e. combinations and dependencies between events, while taking into account the individual points in time at which the events become *True*, and taking into account possible sequences between events. As it includes a concept of time, the temporal logic rules are more extensive and more complex than Boolean algebra.

There are two major differences between the application of the TFTA temporal logic and the Boolean logic:

1. Event sequences are expressed by the order in which events and operators are positioned in a temporal expression; therefore, the laws of commutation, laws of associativity, and distributive laws are not fully applicable.
2. In temporal logic there are logical *contradictions*, i.e. temporal relationships between events that are “not possible”. Such contradictions always yield a logic *False*. For instance, an event can not become *True* after it has already become *True*, and thus $X \vec{\wedge} X = \text{False}$.

4.2.1 Boolean Algebra

The conventional Boolean algebra describes *Boolean relationships* between events, i.e. it makes statements on different events becoming *True*; but it does not take into account the timing between those events. Boolean logic basically consists of the rules listed below [8, 14]:

laws of commutation

$$A \wedge B = B \wedge A \quad \text{and} \quad A \vee B = B \vee A . \quad (4.30)$$

laws of associativity

$$\begin{aligned} A \wedge (B \wedge C) &= (A \wedge B) \wedge C = A \wedge B \wedge C \quad \text{and} \\ A \vee (B \vee C) &= (A \vee B) \vee C = A \vee B \vee C . \end{aligned} \quad (4.31)$$

distributive laws

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \quad \text{and} \quad A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C) . \quad (4.32)$$

laws of idempotency

$$A \wedge A = A \quad \text{and} \quad A \vee A = A . \quad (4.33)$$

laws of absorption

$$A \wedge (A \vee B) = A \quad \text{and} \quad A \vee (A \wedge B) = A . \quad (4.34)$$

de Morgan's theorems

$$\neg(A \wedge B) = \neg A \vee \neg B \quad \text{and} \quad \neg(A \vee B) = \neg A \wedge \neg B . \quad (4.35)$$

operations with *False* and *True*

$$\begin{aligned} \neg \text{False} &= \text{True} , \\ A \wedge \text{False} &= \text{False} \quad \text{and} \quad A \wedge \text{True} = A , \\ A \vee \text{False} &= A \quad \text{and} \quad A \vee \text{True} = \text{True} . \end{aligned} \quad (4.36)$$

4.2.2 Law of Completion

The *law of completion* in (4.37) describes the main relationship between Boolean and temporal operators and fault tree gates, see figure 4.2:

$$A \wedge B = (A \vec{\wedge} B) \vee (A \bar{\wedge} B) \vee (B \vec{\wedge} A) . \quad (4.37)$$

Terms on the right side of (4.37) are mutually exclusive (disjoint).

The SAND connection between *different* events expresses (structurally) dependend failures, which may be interpreted as *common cause failures* (CCF). It can be shown that the expectancy value of the failure probability/failure rate is zero for failure events which are connected by SANDs, if *independent* failures are assumed. For instance, $E[A \bar{\wedge} B] = 0$, see chapter 5.3.1 for details. The SAND operator is also very important for transformations of temporal expressions and for qualitative analysis.

4.2.3 Law of Contradiction

In general, it is logically contradictory if the same event becomes *True* after itself. This follows directly from the assumption of monotony combined with non-repairable components; see chapter 3.2.1 for these two general assumptions of this thesis.

In the most simple case,

$$A \vec{\wedge} A = \text{False} . \quad (4.38)$$

More generally, an event sequence yields *False* if at least one event exists more than once in it; i.e.

$$X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n = \text{False} , \quad (4.39)$$

if $\exists X_i = X_j$ for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. In a temporal fault tree a PAND gate therefore yields *False* if it has the same event as input more than once.

The law of contradiction applies to non-atomic core events analogously:

$$(A \bar{\wedge} B) \vec{\wedge} A = (B \bar{\wedge} A) \vec{\wedge} A = \text{False} , \quad (4.40)$$

$$A \vec{\wedge} (A \bar{\wedge} B) = A \vec{\wedge} (B \bar{\wedge} A) = \text{False} , \quad (4.41)$$

or, more generally,

$$K_1 \vec{\wedge} K_2 \vec{\wedge} \dots \vec{\wedge} K_n = \text{False} , \quad (4.42)$$

if there is at least one atomic event X which is part of two or more core events K , i.e. if $\exists (X \in K_i) \wedge (X \in K_j)$ for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$.

An example: $(A \bar{\wedge} B) \vec{\wedge} C \vec{\wedge} (A \bar{\wedge} D \bar{\wedge} E) = \text{False}$, as $(A \bar{\wedge} B)$ and $(A \bar{\wedge} D \bar{\wedge} E)$ both contain the same atomic event A .

4.2.4 Temporal Law of Idempotency

A new *temporal law of idempotency* may be derived from the laws of completion and the law of contradiction. The temporal law of idempotency applies only to the SAND operator. From (4.37) and (4.38) and the Boolean law of idempotency in (4.33) follows that

$$\begin{aligned} A \wedge A &= (A \vec{\wedge} A) \vee (A \bar{\wedge} A) \vee (A \vec{\wedge} A) = \text{False} \vee (A \bar{\wedge} A) \vee \text{False} \quad \text{and} \\ A \wedge A &= A \quad , \text{ and therefore} \\ A \bar{\wedge} A &= A . \end{aligned} \quad (4.43)$$

4.2.5 Temporal Law of Commutativity

A *temporal law of commutativity* (or *commutation*) applies only to the SAND operator, as

$$A \bar{\wedge} B = B \bar{\wedge} A , \quad (4.44)$$

but not for the PAND operator, as

$$A \vec{\wedge} B \neq B \vec{\wedge} A . \quad (4.45)$$

4.2.6 Temporal Law of Associativity

The SAND operator also has the property of associativity; thus

$$A \bar{\wedge} (B \bar{\wedge} C) = A \bar{\wedge} B \bar{\wedge} C = (A \bar{\wedge} B) \bar{\wedge} C . \quad (4.46)$$

The PAND operator, on the other hand, is only left-associative, as in

$$(A \vec{\wedge} B) \vec{\wedge} C = A \vec{\wedge} B \vec{\wedge} C \neq A \vec{\wedge} (B \vec{\wedge} C) . \quad (4.47)$$

4.2.7 Further Temporal Logic Laws

There are two more temporal laws with special significance:

$$A \vec{\wedge} (B \vec{\wedge} C) = (A \wedge B) \vec{\wedge} C \quad \text{and} \quad (4.48)$$

$$A \bar{\wedge} (B \vec{\wedge} C) = B \vec{\wedge} (A \bar{\wedge} C) . \quad (4.49)$$

Examples illustrating the laws of temporal TFTA logic

The correctness of these two laws is demonstrated using three different graphical methods:

- Table 4.2 (page 36) shows correctness of (4.48) and (4.49) using truth tables similar to the ones known from Boolean logic. The main difference is, that in the temporal logic all possible event sequences have to be taken into account.
- Figure 4.8 shows sequential failure trees for (4.48) and (4.49), see page 34, which are well suited to verify and visualize temporal expressions.
- Finally, figure 4.9 shows the correctness of (4.48) and (4.49) using timing diagrams, see page 35.

The number of entries, i.e. rows, in the truth table equals the number of nodes in the sequential failure tree. Indeed, one can use sequential failure trees in order to simplify the process of creating the truth table. Timing diagrams, on the other hand, are well suited for specific checks of more complex temporal expressions.

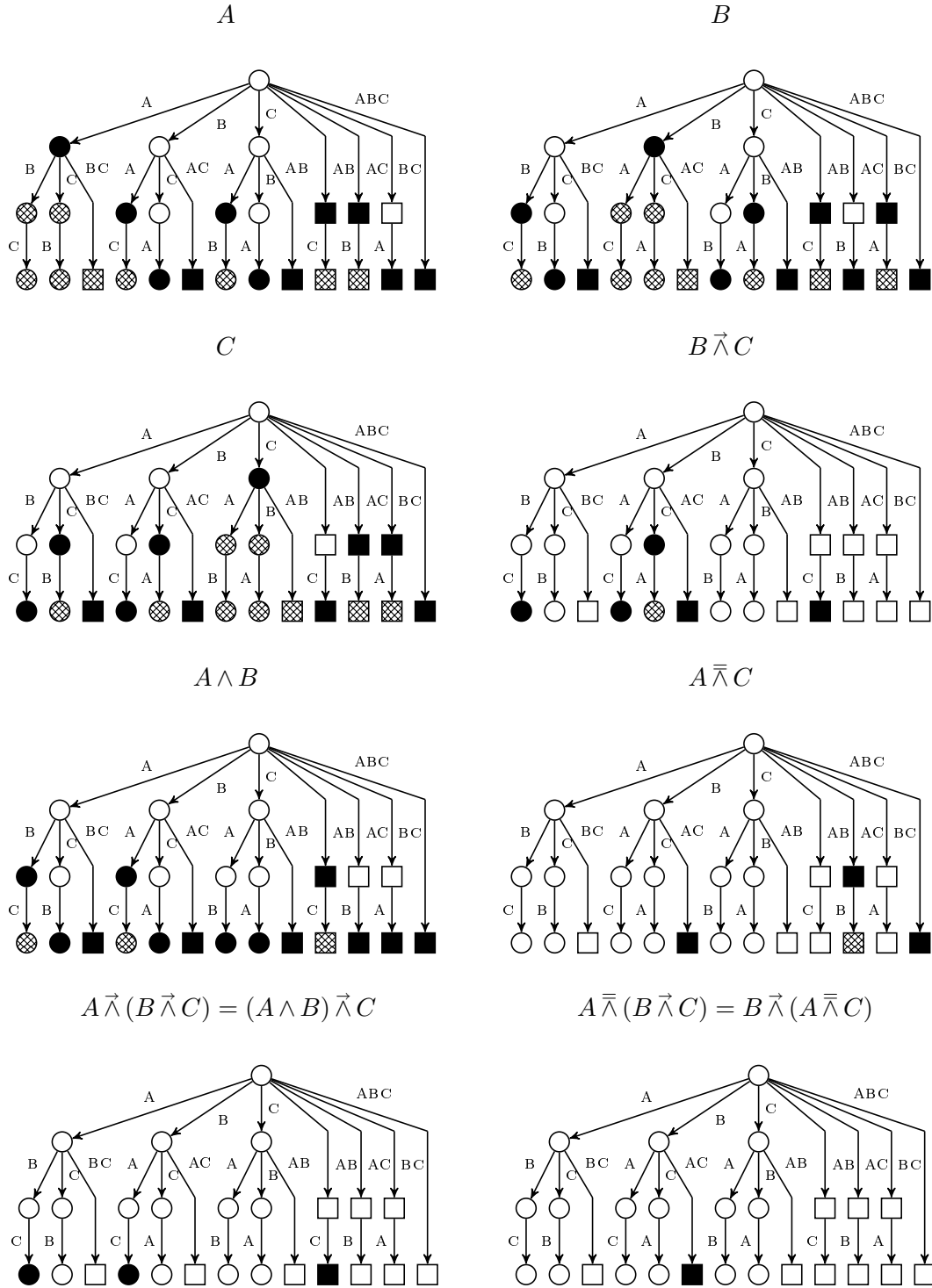


Figure 4.8: Sequential failure trees for (4.48) and (4.49), which show their correctness. From left to right and from top to bottom: A , B , C , $B \vec{\wedge} C$, $A \wedge B$, $A \bar{\wedge} C$, $A \vec{\wedge} (B \vec{\wedge} C) = (A \wedge B) \vec{\wedge} C$, $A \bar{\wedge} (B \vec{\wedge} C) = B \vec{\wedge} (A \bar{\wedge} C)$.

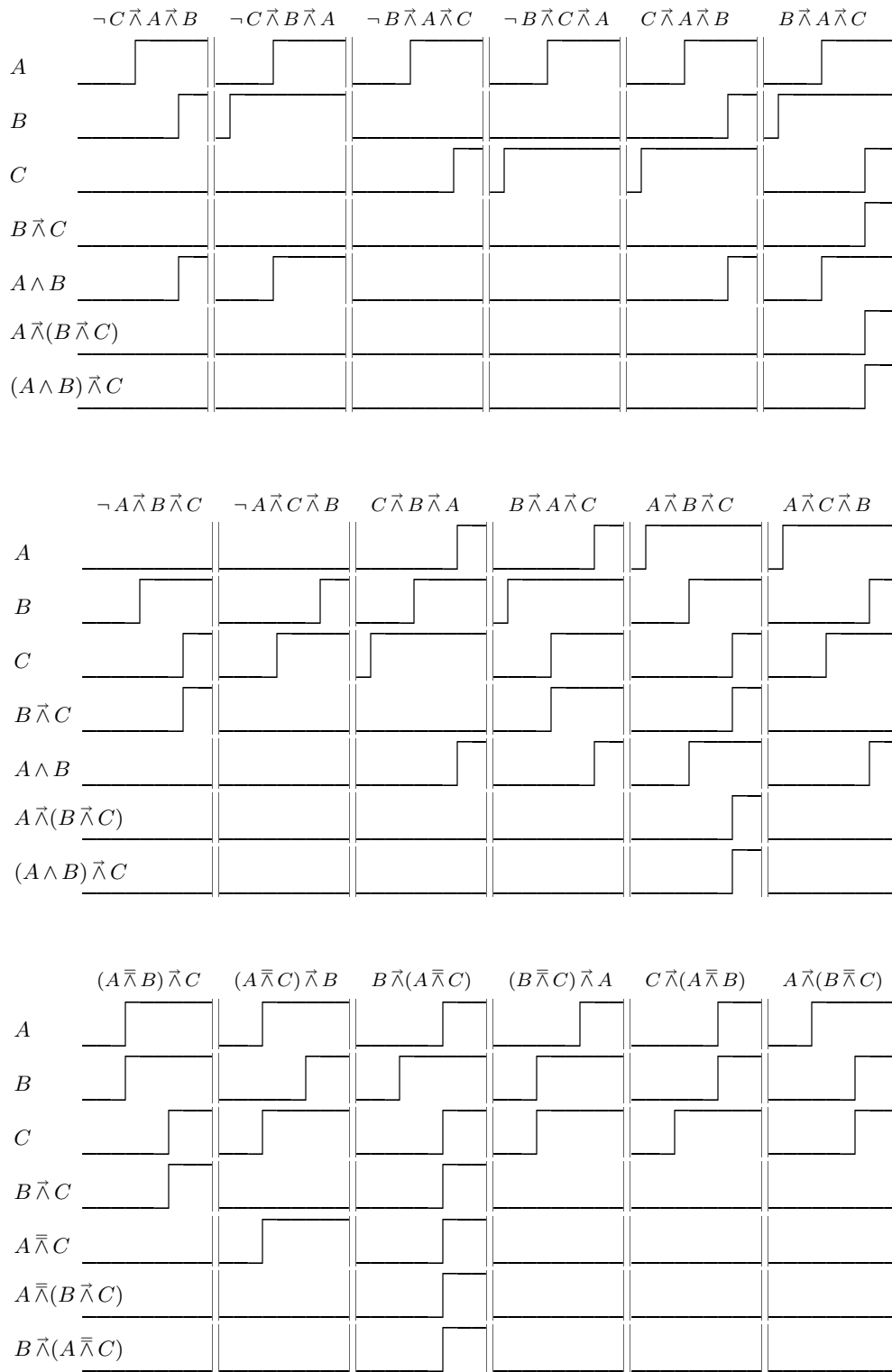


Figure 4.9: Timing diagrams showing selected sequences from table 4.2, which demonstrate the correctness of (4.48) (upper two diagrams) and (4.49) (lower diagram).

$A \vec{\wedge} (B \vec{\wedge} C) \quad (A \wedge B) \vec{\wedge} C$			$A \bar{\wedge} (B \vec{\wedge} C) \quad B \vec{\wedge} (A \bar{\wedge} C)$		
$\neg A \neg B \neg C$	False	False	$\neg A \neg B \neg C$	False	False
$\neg B \neg C \wedge A$	False	False	$\neg B \neg C \wedge A$	False	False
$\neg A \neg C \wedge B$	False	False	$\neg A \neg C \wedge B$	False	False
$\neg A \neg B \wedge C$	False	False	$\neg A \neg B \wedge C$	False	False
$\neg C \wedge (A \vec{\wedge} B)$	False	False	$\neg C \wedge (A \vec{\wedge} B)$	False	False
$\neg C \wedge (B \vec{\wedge} A)$	False	False	$\neg C \wedge (B \vec{\wedge} A)$	False	False
$\neg C \wedge (A \bar{\wedge} B)$	False	False	$\neg C \wedge (A \bar{\wedge} B)$	False	False
$\neg B \wedge (A \vec{\wedge} C)$	False	False	$\neg B \wedge (A \vec{\wedge} C)$	False	False
$\neg B \wedge (C \vec{\wedge} A)$	False	False	$\neg B \wedge (C \vec{\wedge} A)$	False	False
$\neg B \wedge (A \bar{\wedge} C)$	False	False	$\neg B \wedge (A \bar{\wedge} C)$	False	False
$\neg A \wedge (B \vec{\wedge} C)$	False	False	$\neg A \wedge (B \vec{\wedge} C)$	False	False
$\neg A \wedge (C \vec{\wedge} B)$	False	False	$\neg A \wedge (C \vec{\wedge} B)$	False	False
$\neg A \wedge (B \bar{\wedge} C)$	False	False	$\neg A \wedge (B \bar{\wedge} C)$	False	False
$A \vec{\wedge} B \vec{\wedge} C$	True	True	$A \vec{\wedge} B \vec{\wedge} C$	False	False
$B \vec{\wedge} A \vec{\wedge} C$	True	True	$B \vec{\wedge} A \vec{\wedge} C$	False	False
$A \vec{\wedge} C \vec{\wedge} B$	False	False	$A \vec{\wedge} C \vec{\wedge} B$	False	False
$C \vec{\wedge} A \vec{\wedge} B$	False	False	$C \vec{\wedge} A \vec{\wedge} B$	False	False
$B \vec{\wedge} C \vec{\wedge} A$	False	False	$B \vec{\wedge} C \vec{\wedge} A$	False	False
$C \vec{\wedge} B \vec{\wedge} A$	False	False	$C \vec{\wedge} B \vec{\wedge} A$	False	False
$A \vec{\wedge} (B \bar{\wedge} C)$	False	False	$A \vec{\wedge} (B \bar{\wedge} C)$	False	False
$B \vec{\wedge} (A \bar{\wedge} C)$	False	False	$B \vec{\wedge} (A \bar{\wedge} C)$	True	True
$C \vec{\wedge} (A \bar{\wedge} B)$	False	False	$C \vec{\wedge} (A \bar{\wedge} B)$	False	False
$(A \bar{\wedge} B) \vec{\wedge} C$	True	True	$(A \bar{\wedge} B) \vec{\wedge} C$	False	False
$(A \bar{\wedge} C) \vec{\wedge} B$	False	False	$(A \bar{\wedge} C) \vec{\wedge} B$	False	False
$(B \bar{\wedge} C) \vec{\wedge} A$	False	False	$(B \bar{\wedge} C) \vec{\wedge} A$	False	False
$A \bar{\wedge} B \bar{\wedge} C$	False	False	$A \bar{\wedge} B \bar{\wedge} C$	False	False

Table 4.2: Truth table which demonstrates that (4.48) (left side) and (4.49) (right side) are correct. Including SAND connections, there are 26 possible sequences. Logical equivalence of both expressions is shown as in both cases all possible sequences yield identical results.

4.2.8 Temporal Operations with Negated Events

Remark: The statements below exclusively relate to atomic negated events. Specialities of non-atomic negated events are covered from page 40 on.

4.2.8.1 How to Interpret Negated Events in TFTA

In the TFTA, as well as in the conventional FTA, a non-negated event represents a failure of a real element, e.g. a component. Therefore, a negated event represents the “not-failing” of a real element.

There are two possible interpretations for “not-failing”:

1. An element, that has failed before, is repaired. The “not-failing” is an “un-failing”, a transition from one state (failed) to another (repaired), and thus is an action.
2. An element has not yet failed and is still operational. The “not-failing” is a state.

The temporal logic, as discussed in this theses and applied to the TFTA, relies on the assumptions of monotony of the temporal failure function as well as non-repairability of elements.

At first, at time $t = 0$, all elements (components) are operational. Failures occur at times $t > 0$ and are represented in the temporal fault tree by (non-negated) failure events X_i . The latter “switch” from *False* to *True* at times $t_{X_i} > 0$. Moreover, all elements are non-repairable. Failure events that occurred (became *True*) at t_{X_i} stay *True*.

Two things follow for negated events: they are *True* until t_{X_i} and then become *False*; and they cannot become *True* again after t_{X_i} . Thus, a negated event in the TFTA

$$\neg X_i = \begin{cases} \text{True} & \text{in } [0; t_{X_i}[\quad \text{and} \\ \text{False} & \text{in } [t_{X_i}; \infty[, \end{cases} \quad (4.50)$$

with $t_{X_i} > 0$.

Therefore, the first interpretation of the meaning of negated events in the TFTA is to be rejected; in the TFTA negated failure events represent elements, that have not yet failed.

4.2.8.2 Using Negated Events in TFTA

Negated events are used in two different ways within the TFTA; these are comparable to the two ways of using negated events in Boolean FTA.

1. Even if there are no NOT gates used explicitly in the fault tree, the temporal failure function may get negated events from logical transformations. For instance, the transformation of temporal expressions that are not mutually exclusive (not disjoint) into a disjoint form requires usage of negated events.
2. NOT gates in the fault tree model allow explicit modelling of negated events. Such negations of basic events or non-atomic events (subtrees) are then input to other higher-level fault tree gates. Accordingly, the failure function then includes negated events.

Negated Events Resulting From Logic Replacements

In the Boolean FTA non-disjoint expressions are transformed into a disjoint form using negated events [34, 80, 81]. Thereby, negated events only occur within conjunctions (AND connected terms) in combination with at least one non-negated event. The assumption of monotony is not invalidated, because events are not substantially meshed by this transformation (the topic of substantial meshing is discussed in [7]). Moreover, none of the transformation laws of the Boolean logic introduce new negated events – de Morgan’s theorems only discuss transformation of existing negated events.

The temporal logic of the TFTA also uses negated events for the transformation into a disjoint form, see chapter 4.3. But other than the Boolean logic, there are temporal transformation laws, specifically the temporal distributive laws in chapter 4.2.10, that do introduce negated events. These negated events only occur within conjunctions, though, and in combination with at least one non-negated event. In doing so, the assumption of monotony is not invalidated.

Using Negations Explicitly in Fault Trees

This kind of usage of negated events is restricted to cases where no substantially meshed negated events are used in order to not invalidate the assumption of monotony, see [7]. Usually, this is limited to special use cases, e.g. if the results of one of the temporal laws of transformation (see above) shall explicitly be modelled with a temporal fault tree.

In general, TFTA statements like, e.g.,

- “A has not failed yet, before B has not failed yet”, i.e. $\neg A \vec{\wedge} \neg B$, or
- “A and B have simultaneously not failed yet”, i.e. $\neg A \bar{\wedge} \neg B$, or
- “A has failed, because B has not failed yet, or C has failed”, i.e. $A = \neg B \vee C$,

are neither logically meaningful nor allowed in TFTA. Thus there is no necessity to use negated events explicitly as inputs to PAND or SAND gates, or to use them in combination with non-negated events as inputs to OR gates.

On the other hand, it is indeed permitted to model logical statements like $\neg A \wedge B$ explicitly within the fault tree, if – and only if – the assumption of monotony still holds.

4.2.8.3 Rules of Replacement for Negated Events in the Temporal Logic

The law of completion from (4.37) must not be used on expressions where at least one of the operands of the conjunction (AND connection) is a negated event.

Therefore, the application of the other temporal laws of transformation also does not lead to negated events being input to PAND or SAND operators. In case of the temporal distributive laws all negated events are part of conjunction terms, see chapter 4.2.10. Furthermore, this leads to the conclusion that the Boolean logic rules may be used for handling of negated events, see chapter 4.2.1.

Special considerations are necessary for “mixed expressions” where negated events and temporal expressions are both part of the same conjunction. There are

$$\neg A \wedge (\dots \vec{\wedge} A \vec{\wedge} \dots) = \text{False} , \quad (4.51)$$

$$\neg A \wedge (\dots \vec{\wedge} (A \bar{\wedge} \dots) \vec{\wedge} \dots) = \text{False} , \quad (4.52)$$

and

$$(\neg A \wedge B) \wedge C = [\neg A \wedge (B \wedge C)] \vee [(B \vec{\wedge} A) \wedge C] =$$

$$= [\neg A \wedge (B \wedge C)] \vee [B \vec{\wedge} A \vec{\wedge} C] \vee [B \vec{\wedge} (A \vec{\wedge} C)] , \quad (4.53)$$

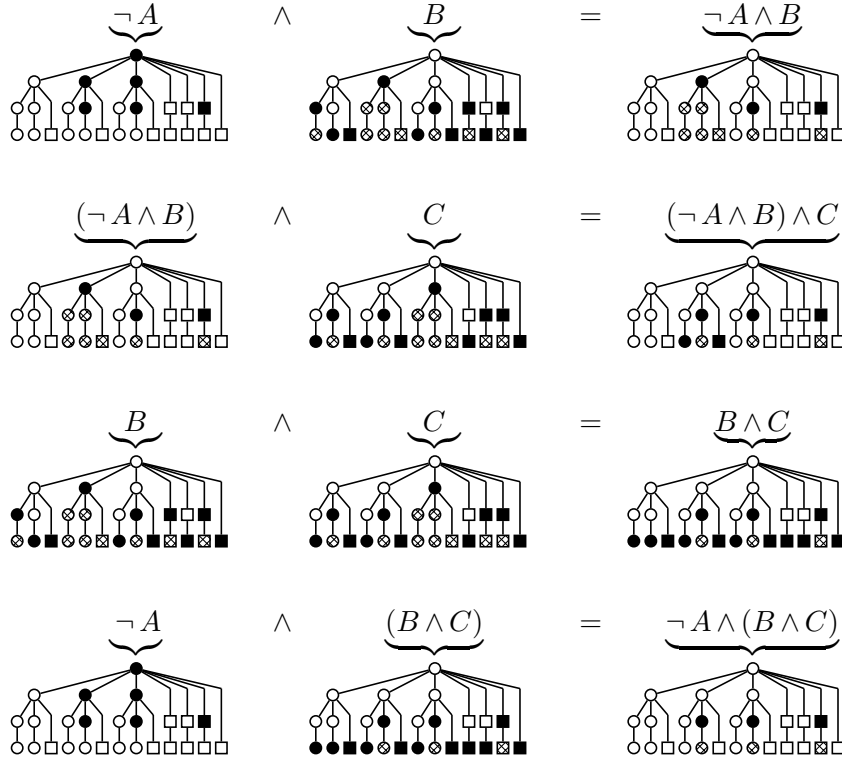
$$(\neg A \wedge B) \vec{\wedge} C = [\neg A \wedge (B \vec{\wedge} C)] \vee [B \vec{\wedge} A \vec{\wedge} C] \vee [B \vec{\wedge} (A \vec{\wedge} C)] , \quad (4.54)$$

$$(\neg A \wedge B) \vec{\wedge} C = \neg A \wedge (B \vec{\wedge} C) , \quad (4.55)$$

$$C \vec{\wedge} (\neg A \wedge B) = \neg A \wedge (C \vec{\wedge} B) . \quad (4.56)$$

Equation (4.53) shows the one main difference between temporal and Boolean logic with regards to usage of negated events.

In the Boolean logic the law of associativity from equation (4.31) also applies to negated events. But in the temporal logic negated events have a “period of validity”, which is expressed by brackets. For instance, $(\neg A \wedge B) \wedge C$ denotes two things: first, that at the point in time, at which event B occurs, event A has “not yet” occurred, and second, that C has occurred; but there is no separate statement on the timing relationship between C and the others. On the other hand, $\neg A \wedge (B \wedge C)$ expresses timing relationships between all three events; this expression denotes that at the point in time, at which “ B and C ” occurs, the event A has not yet occurred:



In particular, this also affects temporal expressions of the following type:

$$(\neg A \wedge B) \vec{\wedge} A = \neg A \wedge (B \vec{\wedge} A) = \text{False} , \quad (4.57)$$

$$A \vec{\wedge} (\neg A \wedge B) = \neg A \wedge (A \vec{\wedge} B) = \text{False} , \quad (4.58)$$

$$\begin{aligned} (\neg A \wedge B) \vec{\wedge} A &= [\neg A \wedge (B \vec{\wedge} A)] \vee [B \vec{\wedge} A \vec{\wedge} A] \vee [B \vec{\wedge} (A \vec{\wedge} A)] = \\ &= \text{False} \vee \text{False} \vee [B \vec{\wedge} A] = B \vec{\wedge} A . \end{aligned} \quad (4.59)$$

Chapter 4.3.2.2 discusses why and how these expressions are “temporally (non-)minimal”.

4.2.8.4 Conjunction of Negated Events

The above discussion did not include conjunctions consisting of more than one negated event, as e.g. in

$$\neg A \neg B = \neg A \wedge \neg B . \quad (4.60)$$

When applied to the TFTA, such conjunctions are interpreted as undividable entities; the rules for transformation and handling of negated events, as given above, apply to those entities analogously.

From this follows that

$$\neg A \wedge (\neg B \wedge C) = (\neg A \neg B) \wedge C . \quad (4.61)$$

4.2.8.5 Temporal Laws of Negation, i.e. Negation of Non-Atomic Negated Events

So far, all statements regarding negated events have applied to atomic events (basic events) only. Additional aspects have to be considered in case of negated non-atomic events, as e.g. in $\neg(A \vec{\wedge} B)$.

The Negation of Boolean non-atomic expressions like $\neg(A \wedge B)$ or $\neg(A \vee B)$ is done using de Morgan's theorems in (4.35). The negation of SAND and PAND connected expressions can, for example, be deduced from figure 4.2; it yields:

$$\neg(A \vec{\wedge} B) = (\neg A \neg B) \vee (\neg B \wedge A) \vee (\neg A \wedge B) \vee (B \vec{\wedge} A) \vee (A \bar{\wedge} B) \quad \text{and} \quad (4.62)$$

$$\neg(A \bar{\wedge} B) = (\neg A \neg B) \vee (\neg B \wedge A) \vee (\neg A \wedge B) \vee (A \vec{\wedge} B) \vee (B \vec{\wedge} A) . \quad (4.63)$$

On the right hand side of the equations all terms are mutually exclusive (disjoint) and carry explicite (temporal) statements to all events involved, see chapter 4.3.3.

In TFTA such non-atomic negated expressions can only exist as part of a conjunction expression together with non-negated events. As such, they describe a system state where at a specific point in time a specific event sequence has “not yet” occurred. The right hand sides of (4.35) and (4.62) and (4.63) represent the different possibilities how this specific system state was reached.

An example: the temporal expressions $\neg(A \vec{\wedge} B) \wedge C$ represents a state in which at the time of occurrence of C the event sequence $A \vec{\wedge} B$ has not occurred. This implies either that at the time of occurrence of C

- neither A nor B have occurred – therefore $(\neg A \neg B) \wedge C$ –
- or A has occurred, but B has not – therefore $\neg B \wedge (A \wedge C)$ –
- or B has occurred, but A has not – therefore $\neg A \wedge (B \wedge C)$ –
- or B has occurred before A has occurred – therefore $(B \vec{\wedge} A) \wedge C$ –
- or A and B have occurred simultaneously – therefore $(A \bar{\wedge} B) \wedge C$.

The *first temporal law of negation* is thus given as

$$\begin{aligned} \neg(A \bar{\wedge} B) \wedge C = & [(\neg A \neg B) \wedge C] \vee [\neg B \wedge (A \wedge C)] \vee [\neg A \wedge (B \wedge C)] \vee \\ & \vee [(A \vec{\wedge} B) \wedge C] \vee [(B \vec{\wedge} A) \wedge C] . \end{aligned} \quad (4.64)$$

Analogously, the *second temporal law of negation* is given as

$$\begin{aligned} \neg(A \vec{\wedge} B) \wedge C = & [(\neg A \neg B) \wedge C] \vee [\neg B \wedge (A \wedge C)] \vee [\neg A \wedge (B \wedge C)] \vee \\ & \vee [(B \vec{\wedge} A) \wedge C] \vee [(A \bar{\wedge} B) \wedge C] . \end{aligned} \quad (4.65)$$

4.2.9 True and False in Temporal Logics

Operations with the “timeless” expressions *True* and *False* should only be found in TFTA expressions, if a more complex temporal expression was reduced to *True* or *False* in a preceeding transformation step.

If X and $X \neq \text{True}$ themselves are not negated, then

$$X \vec{\wedge} \text{True} = \text{False} , \quad X \bar{\wedge} \text{True} = \text{False} , \quad \text{True} \vec{\wedge} X = X , \quad (4.66)$$

$$X \vec{\wedge} \text{False} = \text{False} , \quad X \bar{\wedge} \text{False} = \text{False} , \quad \text{False} \vec{\wedge} X = \text{False} . \quad (4.67)$$

Furthermore,

$$\text{True} \vec{\wedge} \text{True} = \text{False} , \quad \text{True} \bar{\wedge} \text{True} = \text{True} , \quad \text{False} \vec{\wedge} \text{True} = \text{False} . \quad (4.68)$$

Given these rules, consistency to the Boolean logic rules, which are, of course, still valid, is obtained; thus,

$$\begin{aligned} X \wedge \text{True} &= (X \vec{\wedge} \text{True}) \vee (X \bar{\wedge} \text{True}) \vee (\text{True} \vec{\wedge} X) = \text{True} \vec{\wedge} X = X , \\ X \wedge \text{False} &= (X \vec{\wedge} \text{False}) \vee (X \bar{\wedge} \text{False}) \vee (\text{False} \vec{\wedge} X) = \text{False} \vec{\wedge} X = \text{False} . \end{aligned}$$

4.2.10 Temporal Distributive Laws

Boolean logic has the distributive law as given in (4.32). Combined with the Boolean operators' property of associativity, see (4.31), this yields

$$(A \vee B) \wedge C = C \wedge (B \vee A) = (A \wedge C) \vee (B \wedge C) = (C \wedge B) \vee (C \wedge A) . \quad (4.69)$$

This distributive law is vital to the transformation of Boolean expressions into a disjunctive normal form (DNF).

Very similar, the SAND operator of the temporal logic also has the property of associativity; therefore, the temporal laws of associativity and commutativity apply, see (4.44) and (4.46).

On the other hand, the PAND operator obviously lacks a law of commutativity, see (4.45); reason for that is that this operator “transports” a great part of its logic information in the sequence of events.

Therefore, at least the following has to be differentiated for something like a PAND's distributive law:

$$A \vec{\wedge} (B \vee C) , \text{ so-called } \textit{type I}, \quad \text{and} \quad (4.70)$$

$$(A \vee B) \vec{\wedge} C , \text{ so-called } \textit{type II}. \quad (4.71)$$

The following two sections discuss temporal distributive laws, first for PAND operators and expressions of type I and II, followed by the temporal distributive law for SAND operators; for the latter, no further discrimination of types is necessary.

4.2.10.1 Distributive Law for PAND-OR Expressions of Type I

The logic statment of expression $A \vec{\wedge} (B \vee C)$ is: “ A must occur, before the expression in brackets ($B \vee C$) occurs”. This is *not* equivalent to the logic statement “ A must occur before B , or A must occur before C ”, as proven by table 4.3 and figure 4.10:

$$A \vec{\wedge} (B \vee C) \neq (A \vec{\wedge} B) \vee (A \vec{\wedge} C) , \quad (4.72)$$

and thus there is no simple temporal distributive law for expressions of type I.

In fact, the expression on the left hand side of (4.72) makes no explicit statement on temporal dependencies between events B and C ; but it does include an implicit temporal dependency between B and C . This temporal dependency not so much affects the occurrence of (further) events, but the non-occurrence of $A \vec{\wedge} (B \vee C)$ if one of the events B or C occurs before A . This implicit dependency is lost in the right hand side of (4.72).

This problem is solved by explicitly stating the temporal dependencies which are only implied by the left side of (4.72).

The relevant expressions $(B \vee C)$ splits into five possible sequences:

$$(B \vee C) = (\neg C \vec{\wedge} B) \vee (\neg B \vec{\wedge} C) \vee (B \vec{\wedge} C) \vee (C \vec{\wedge} B) \vee (B \bar{\wedge} C) .$$

Only three of these sequences are minimal failure sequences, see figure 4.10 (left side):

$$(B \vee C) = (\neg C \wedge B) \vee (\neg B \wedge C) \vee (B \bar{\wedge} C) . \quad (4.73)$$

Inserting this into (4.70) yields for temporal expressions of type I, that

$$A \vec{\wedge} (B \vee C) = A \vec{\wedge} [(\neg C \wedge B) \vee (\neg B \wedge C) \vee (B \bar{\wedge} C)] . \quad (4.74)$$

At this point non-minimal sequences need not be considered. The OR connected terms in brackets are on the right hand side of the PAND operator, and thus occur “later”; all non-minimal terms then occur “later still”. They are covered by the minimal sequences.

Now, with all temporal dependencies explicitly stated, a distribution of the expression is possible, thus

$$A \vec{\wedge} (B \vee C) = [A \vec{\wedge} (\neg C \wedge B)] \vee [A \vec{\wedge} (\neg B \wedge C)] \vee [A \vec{\wedge} (B \bar{\wedge} C)] . \quad (4.75)$$

	$A \vec{\wedge} (B \vee C)$ $(A \vec{\wedge} B) \vee (A \vec{\wedge} C)$			$A \vec{\wedge} (B \vee C)$ $(A \vec{\wedge} B) \vee (A \vec{\wedge} C)$	
$\neg A \neg B \neg C$	<i>False</i>	<i>False</i>	$A \vec{\wedge} B \vec{\wedge} C$	<i>True</i>	<i>True</i>
$\neg B \neg C \wedge A$	<i>False</i>	<i>False</i>	$B \vec{\wedge} A \vec{\wedge} C$	<i>False</i>	<i>True</i>
$\neg A \neg C \wedge B$	<i>False</i>	<i>False</i>	$A \vec{\wedge} C \vec{\wedge} B$	<i>True</i>	<i>True</i>
$\neg A \neg B \wedge C$	<i>False</i>	<i>False</i>	$C \vec{\wedge} A \vec{\wedge} B$	<i>False</i>	<i>True</i>
$\neg C \wedge (A \vec{\wedge} B)$	<i>True</i>	<i>True</i>	$B \vec{\wedge} C \vec{\wedge} A$	<i>False</i>	<i>False</i>
$\neg C \wedge (B \vec{\wedge} A)$	<i>False</i>	<i>False</i>	$C \vec{\wedge} B \vec{\wedge} A$	<i>False</i>	<i>False</i>
$\neg C \wedge (A \bar{\wedge} B)$	<i>False</i>	<i>False</i>	$A \vec{\wedge} (B \bar{\wedge} C)$	<i>True</i>	<i>True</i>
$\neg B \wedge (A \vec{\wedge} C)$	<i>True</i>	<i>True</i>	$B \vec{\wedge} (A \bar{\wedge} C)$	<i>False</i>	<i>False</i>
$\neg B \wedge (C \vec{\wedge} A)$	<i>False</i>	<i>False</i>	$C \vec{\wedge} (A \bar{\wedge} B)$	<i>False</i>	<i>False</i>
$\neg B \wedge (A \bar{\wedge} C)$	<i>False</i>	<i>False</i>	$(A \bar{\wedge} B) \vec{\wedge} C$	<i>False</i>	<i>True</i>
$\neg A \wedge (B \vec{\wedge} C)$	<i>False</i>	<i>False</i>	$(A \bar{\wedge} C) \vec{\wedge} B$	<i>False</i>	<i>True</i>
$\neg A \wedge (C \vec{\wedge} B)$	<i>False</i>	<i>False</i>	$(B \bar{\wedge} C) \vec{\wedge} A$	<i>False</i>	<i>False</i>
$\neg A \wedge (B \bar{\wedge} C)$	<i>False</i>	<i>False</i>	$A \bar{\wedge} B \bar{\wedge} C$	<i>False</i>	<i>False</i>

Table 4.3: Truth table for expressions $A \vec{\wedge} (B \vee C)$ and $(A \vec{\wedge} B) \vee (A \vec{\wedge} C)$. Including SANDs there are 26 sequences, which are divided into two groups of 13 each. As both expressions do not yield same results for all sequences (see deviations in bold), both expressions are not equivalent.

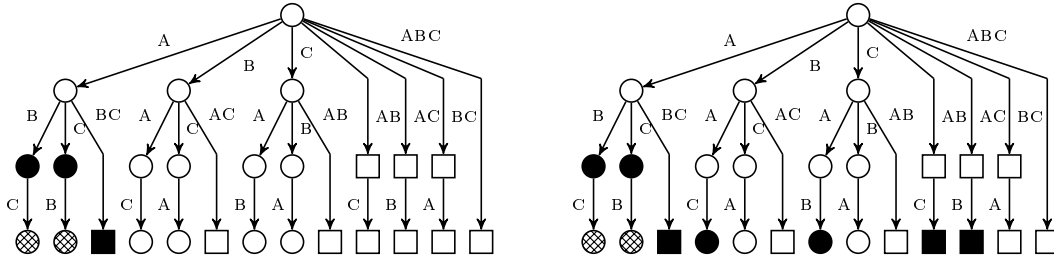


Figure 4.10: Left side: Sequential failure tree for expression $A \bar{\wedge} (B \vee C)$. Right side: Sequential failure tree for expression $(A \bar{\wedge} B) \vee (A \bar{\wedge} C)$. On the right side there are additional sequences, as each of the two sub-expressions $(A \bar{\wedge} B)$ and $(A \bar{\wedge} C)$ does not make any statements about the occurrence of the missing third event.

Further transformation of this according to chapter 4.2.8 then leads to the distributive law for temporal expression of type I:

$$A \bar{\wedge} (B \vee C) = [\neg C \wedge (A \bar{\wedge} B)] \vee [\neg B \wedge (A \bar{\wedge} C)] \vee [A \bar{\wedge} (B \bar{\wedge} C)] . \quad (4.76)$$

The distributive law for temporal expression of type I therefore requires explicit statements on the (non-)occurrence of all of the relevant events, and requires such statements in every sub-expression which is OR connected. Statements with that property are called *temporal minterms* in analogy to Boolean *minterms*.

If the temporal laws of negation are applied, (4.76) holds for the case of non-atomic events A, B, C , too.

Terms on the right side of (4.76) are mutually exclusive (disjoint). This simplifies later probabilistic quantification, see chapter 5.

Simplification if Terms are Disjoint

The relationship in (4.76) also holds for the special case of disjoint events B and C , i.e. $B \perp C$. But $B \perp C$ implies that each of the events B or C occurs only if the other event does not occur and has not yet occurred. Then, (4.76) may be simplified to

$$A \bar{\wedge} (B \vee C) = [A \bar{\wedge} B] \vee [A \bar{\wedge} C] , \quad (4.77)$$

if $B \perp C$.

4.2.10.2 Distributive Law for PAND-OR Expressions of Type II

The logic statement of expression $(A \vee B) \bar{\wedge} C$ is: “the expression in brackets $(A \vee B)$ must occur before C occurs”. This is equivalent to the logic statement “ A must occur before C , or B must occur before C ”, as proven by the sequential failure trees in figure 4.11, which correspond to the three expressions $(A \vee B) \bar{\wedge} C$, $(A \bar{\wedge} C)$, and $(B \bar{\wedge} C)$.

Therefore, the distributive law for temporal expressions of type II is given as

$$(A \vee B) \bar{\wedge} C = (A \bar{\wedge} C) \vee (B \bar{\wedge} C) . \quad (4.78)$$

On the other hand, figure 4.11 also shows that $(A \bar{\wedge} C)$ and $(B \bar{\wedge} C)$ are not mutually exclusive. The joint sequences, which are part of both expressions, are easily found by building the intersection, thus

$$(A \bar{\wedge} C) \wedge (B \bar{\wedge} C) = (A \wedge B) \bar{\wedge} C = [A \bar{\wedge} B \bar{\wedge} C] \vee [B \bar{\wedge} A \bar{\wedge} C] \vee [(A \bar{\wedge} B) \bar{\wedge} C] .$$

Figure 4.11 denotes these sequences with \star .

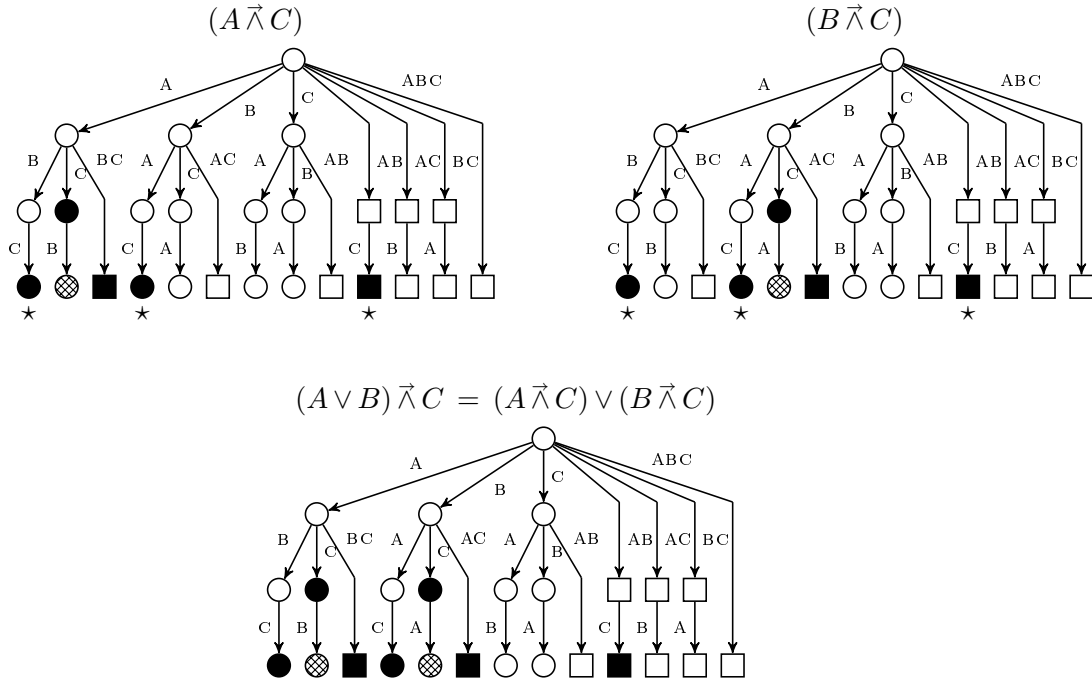


Figure 4.11: Distributive law for temporal expressions of type II: the sequential failure trees of $(A \vee B) \vec{\wedge} C$, $(A \vec{\wedge} C)$, and $(B \vec{\wedge} C)$ show that $(A \vec{\wedge} C)$ and $(B \vec{\wedge} C)$ are minimal but not mutually exclusive (disjoint); joint sequences are marked with \star .

4.2.10.3 Distributive Law for SAND-OR Expressions

The logic statement of expression $A \bar{\wedge} (B \vee C)$ is: “A must occur simultaneously with the expression in brackets $(B \vee C)$ ”. In analogy to the distributive law for temporal expressions of type I it is easily shown that this is *not* equivalent to the logic statement “A occurs simultaneously with B, or A occurs simultaneously with C”, as proven by figure 4.12. In consequence, there is also no simple temporal distributive law for SAND-OR expressions.

Instead, the temporal distributive law for SAND-OR expressions looks similar to (4.76) and is given as

$$A \bar{\wedge} (B \vee C) = [\neg C \wedge (A \bar{\wedge} B)] \vee [\neg B \wedge (A \bar{\wedge} C)] \vee [A \bar{\wedge} B \bar{\wedge} C] . \quad (4.79)$$

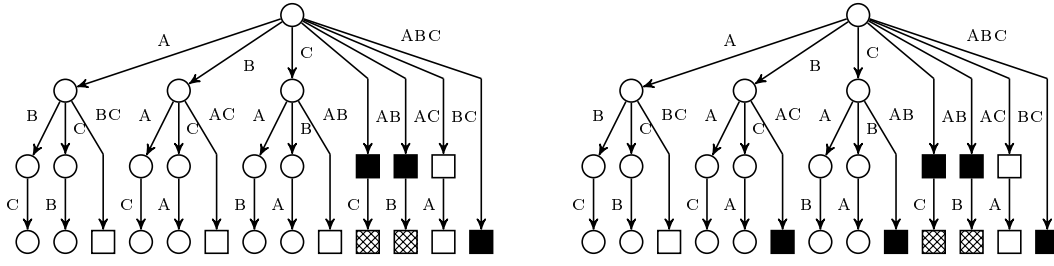


Figure 4.12: Left side: Sequential failure tree for expression $A \bar{\bar{B}}(B \vee C)$. Right side: Sequential failure tree for expression $(A \bar{\bar{B}}) \vee (A \bar{\bar{C}})$. On the right side there are additional sequences, as each of the two sub-expressions $(A \bar{\bar{B}})$ and $(A \bar{\bar{C}})$ does not make any statements about the occurrence of the missing third event.

Simplification if Terms are Disjunct

The relationship in (4.79) also holds for the special case of disjoint events B and C , i.e. $B \perp C$. But $B \perp C$ implies that each of the events B or C occurs only if the other event does not occur and has not yet occurred. Then, (4.79) may be simplified to

$$A \bar{\bar{B}}(B \vee C) = [A \bar{\bar{B}}] \vee [A \bar{\bar{C}}] . \quad (4.80)$$

if $B \perp C$.

4.2.11 Temporal Laws of Absorption

In analogy to the Boolean laws of absorption in (4.34), there are *temporal laws of absorption*, as well. Initially, it may seem that there are several temporal laws of absorption for different numbers of events involved; this intuition come mainly from the permutations that need to be taken into account when analysing event sequences. On the other hand, it can be shown that the temporal laws of absorption really are specializations of the Boolean laws of absorption in (4.34):

Starting with the most simple case with only two events involved, the temporal laws of absorption may be derived from (4.34) by using the law of completion in (4.37); this yields

$$A \vee (A \wedge B) = A = A \vee [(A \bar{\bar{B}}) \vee (A \bar{\bar{B}}) \vee (B \bar{\bar{A}})] , \quad (4.81)$$

which may then be further transformed into

$$A \vee (A \bar{\bar{B}}) = A , \quad (4.82)$$

$$A \vee (B \bar{\bar{A}}) = A , \quad (4.83)$$

$$A \vee (A \bar{\bar{B}}) = A . \quad (4.84)$$

The more “general” event A absorbs the more “concrete” event, if the latter is a subset of A ; this is the same for Boolean and temporal logic. In general, if ES is an (extended) event sequence, then

$$A \vee ES = A , \quad \text{if } A \in ES, \text{ i.e. } ES \subseteq A . \quad (4.85)$$

This relation also holds for non-atomic events A . Other than in the Boolean logic, with more complex temporal expressions it is increasingly difficult to spot subsets. There are two major

reasons for that: the PAND operator has no law of commutativity; and the invention of core events allows for nested events.

For instance, temporal law of absorption for three events are given as

$$(A \vec{\wedge} B) \vee (A \vec{\wedge} B \vec{\wedge} C) = A \vec{\wedge} B, \quad (4.86)$$

$$(A \vec{\wedge} B) \vee (A \vec{\wedge} C \vec{\wedge} B) = A \vec{\wedge} B, \quad (4.87)$$

$$(A \vec{\wedge} B) \vee (C \vec{\wedge} A \vec{\wedge} B) = A \vec{\wedge} B, \quad (4.88)$$

$$(A \vec{\wedge} B) \vee ((A \vec{\wedge} C) \vec{\wedge} B) = A \vec{\wedge} B, \quad (4.89)$$

$$(A \vec{\wedge} B) \vee ((A \vec{\wedge} B) \vec{\wedge} C) = (A \vec{\wedge} B) \vee (A \vec{\wedge} (B \vec{\wedge} C)) = A \vec{\wedge} B. \quad (4.90)$$

Indeed, (4.86) to (4.90) are simple reformulations of

$$(A \vec{\wedge} B) \vee ((A \vec{\wedge} B) \wedge C) = (A \vec{\wedge} B), \quad (4.91)$$

as demonstrated by the following transformation:

$$\begin{aligned} (A \vec{\wedge} B) \wedge C &= [(A \vec{\wedge} B) \vec{\wedge} C] \vee [(A \vec{\wedge} B) \vec{\wedge} \bar{C}] \vee [C \vec{\wedge} (A \vec{\wedge} B)] = \\ &= [A \vec{\wedge} B \vec{\wedge} C] \vee [A \vec{\wedge} (B \vec{\wedge} \bar{C})] \vee [A \vec{\wedge} C \vec{\wedge} B] \vee [C \vec{\wedge} A \vec{\wedge} B] \vee [(A \vec{\wedge} \bar{C}) \vec{\wedge} B]. \end{aligned} \quad (4.92)$$

Taking this concept one step further, the general temporal laws of absorption may then be given in complete analogy to its Boolean counterpart as

$$ES_i \vee ES_j = ES_i \quad \text{for } ES_j \subseteq ES_i. \quad (4.93)$$

The same holds true for the second Boolean law of absorption from (4.34); its temporal version reads as

$$A \vec{\wedge} (A \vee B) = [\neg B \wedge (A \vec{\wedge} A)] \vee [\neg A \wedge (A \vec{\wedge} B)] \vee [A \vec{\wedge} (A \vec{\wedge} B)] = \text{False}, \quad (4.94)$$

$$(A \vee B) \vec{\wedge} A = (A \vec{\wedge} A) \vee (B \vec{\wedge} A) = B \vec{\wedge} A, \quad (4.95)$$

$$\begin{aligned} A \vec{\wedge} (A \vee B) &= [\neg B \wedge (A \vec{\wedge} A)] \vee [\neg A \wedge (A \vec{\wedge} B)] \vee [A \vec{\wedge} (A \vec{\wedge} B)] = \\ &= (\neg B \wedge A) \vee (A \vec{\wedge} B). \end{aligned} \quad (4.96)$$

Although initially not very intuitive, these results are correct, as demonstrated by the following transformation: On the one hand,

$$[A \vec{\wedge} (A \vee B)] \vee [(A \vee B) \vec{\wedge} A] \vee [A \vec{\wedge} (A \vee B)] = A \wedge (A \vee B) = A.$$

And on the other hand, (4.94) to (4.96) yield

$$[A \vec{\wedge} (A \vee B)] \vee [(A \vee B) \vec{\wedge} A] \vee [A \vec{\wedge} (A \vee B)] = (B \vec{\wedge} A) \vee (\neg B \wedge A) \vee (A \vec{\wedge} B).$$

Furthermore, $\neg B \wedge A$ covers the non-minimal sequence $A \vec{\wedge} B$, thus providing

$$\begin{aligned} [A \vec{\wedge} (A \vee B)] \vee [(A \vee B) \vec{\wedge} A] \vee [A \vec{\wedge} (A \vee B)] &= (B \vec{\wedge} A) \vee (\neg B \wedge A) \vee (A \vec{\wedge} B) = \\ &= (\neg B \wedge A) \vee (A \vec{\wedge} B) \vee (A \vec{\wedge} B) \vee (B \vec{\wedge} A) = (\neg B \wedge A) \vee (A \wedge B) = A. \end{aligned}$$

These transformations illustrate that (4.94) to (4.96) really are only specializations of the Boolean laws of absorption.

4.2.12 Temporal Law for Intersections

The introduction of PAND and SAND operators into the temporal TFTA logic leads to expressions like $A \wedge (A \vec{\wedge} B)$, $B \wedge (A \vec{\wedge} B)$, or $A \wedge (A \bar{\wedge} B)$. Such expressions are not easily covered by the temporal laws of absorption, as in their case, and other than in case of the laws of absorption, see above, the more “general” expression does not absorb the more “concrete” expression. Therefore, a new *temporal law for intersections* is proposed.

The temporal law for intersections describes conjunctions of two expression, one of which is an intersection of the other. In the Boolean case, this can be solved by applying the laws of associativity and idempotency:

$$A \wedge (A \wedge B) = A \wedge A \wedge B = A \wedge B . \quad (4.97)$$

In the temporal case, three different settings have to be considered:

$$A \wedge (A \vec{\wedge} B) = (A \vec{\wedge} B) \wedge A = A \vec{\wedge} B , \quad (4.98)$$

$$B \wedge (A \vec{\wedge} B) = (A \vec{\wedge} B) \wedge B = A \vec{\wedge} B , \quad (4.99)$$

$$A \wedge (A \bar{\wedge} B) = (A \bar{\wedge} B) \wedge A = (B \bar{\wedge} A) \wedge A = A \wedge (B \bar{\wedge} A) = A \bar{\wedge} B . \quad (4.100)$$

Correctness may be easily demonstrated using the temporal logic laws provided above. For instance,

$$\begin{aligned} A \wedge (A \vec{\wedge} B) &= [A \vec{\wedge} (A \vec{\wedge} B)] \vee [A \bar{\wedge} (A \vec{\wedge} B)] \vee [(A \vec{\wedge} B) \vec{\wedge} A] = \\ &= (A \wedge A) \vec{\wedge} B \vee False \vee False = A \vec{\wedge} B . \end{aligned}$$

The same holds true for more general cases with more complex expressions, as in

$$X_i \wedge \dots \wedge X_j \wedge (\dots \vec{\wedge} X_i \vec{\wedge} \dots) = X_j \wedge (\dots \vec{\wedge} X_i \vec{\wedge} \dots) \text{ and} \quad (4.101)$$

$$X_i \wedge \dots \wedge X_j \wedge (\dots \bar{\wedge} X_i \bar{\wedge} \dots) = X_j \wedge (\dots \bar{\wedge} X_i \bar{\wedge} \dots) , \quad (4.102)$$

as well as for expressions that include intersections with non-atomic core events, i.e.

$$X_i \wedge \dots \wedge X_j \wedge (\dots \vec{\wedge} (X_i \bar{\wedge} \dots) \vec{\wedge} \dots) = (\dots \vec{\wedge} X_j \wedge (X_i \bar{\wedge} \dots) \vec{\wedge} \dots) . \quad (4.103)$$

In general, the temporal law for intersections is therefore given as:

$$ES_i \wedge ES_j = ES_j \quad \text{for } ES_j \subseteq ES_i . \quad (4.104)$$

4.3 Minimal and Disjoint Forms of TFTA Temporal Expressions

4.3.1 Minimal and Disjoint Forms of Boolean Expressions

This chapter discusses two properties that TFTA temporal expressions may have. Temporal expressions which are minimal or mutually exclusive (disjoint) have special meaning and importance within the TFTA’s temporal logic; in this they are similar to the Boolean FTA. In both cases, the Boolean as well as the temporal, any logic expression can be transformed into “sum of product” forms, i.e. DNF or TDNF, respectively, by using the laws of transformation given in chapter 4.2.

In general, these cutsets (Boolean case) or event sequences (temporal logic) still include redundant information. Therefore, further transformation into a minimal sum of products form, i.e. minimal cutsets and MCSS, respectively, is necessary and provides an even more useful representation of the (temporal) failure function.

For further probabilistic calculation it is then helpful to transform this minimal form into a minterm form, where all minterms are mutually exclusive (disjoint), see chapter 4.3.3.

Disjunctive Normal Form (Sum of Products)

Boolean expressions φ are transformed into a DNF by applying the laws of Boolean algebra; in DNF

$$\varphi = \bigvee_{j=1}^{\zeta} S_j = \bigvee_{j=1}^{\zeta} \left(\bigwedge_{i=1}^{n_j} X_{j,i} \right), \quad (4.105)$$

where ζ denotes the number of *cutsets* S of φ , which are not necessarily already minimal, and n_j denotes the number of events X which constitute S_j .

Minimal DNF

In a next step, the cutsets S of Boolean expressions φ are minimal, if none of the cutsets “includes” another. If so, they are called *minimal cutsets* and are denoted with MS for better discrimination. Using the laws of Boolean algebra from chapter 4.2.1, (monotone) Boolean expressions as in (4.105) can be transformed into a minimal form, where

$$\varphi = \bigvee_{j=1}^{\xi} MS_j = \bigvee_{j=1}^{\xi} \left(\bigwedge_{i=1}^{n_j} X_{j,i} \right), \quad (4.106)$$

where $\xi \leq \zeta$.

Each of these ξ minimal cutsets MS_j and $MS_{j'}$ with $j, j' \in \{1, 2, \dots, \xi\}$ and $j' \neq j$ are pairwise mutually exclusive:

$$MS_j \wedge MS_{j'} \neq MS_j \quad \text{und} \quad MS_j \wedge MS_{j'} \neq MS_{j'}. \quad (4.107)$$

Simplifying Quantification By Using Disjoint Terms

In many cases it is helpful to transform logic functions into a equivalent form which is specifically well suited for a certain task. For conventional fault trees the minimal cutset form of a system's failure function according to (4.106) is, for example, especially illustrative and well suited for qualitative analyses; on the other hand, the form below is equivalent but much less easy to understand:

$$\varphi = \bigvee_{j=1}^{\xi} \left(MS_j \cdot \bigwedge_{i=1}^{j-1} \neg (MS_i) \right). \quad (4.108)$$

This form aids probabilistic analyses because of its mutually exclusive (disjoint) OR connected terms; see chapter 5 for details.

In general, two Boolean expressions φ_1 and φ_2 are mutually exclusive (disjoint), if their conjunction yields *False*:

$$\varphi_1 \wedge \varphi_2 = \text{False} \quad \Longleftrightarrow \quad \varphi_1 \perp \varphi_2. \quad (4.109)$$

4.3.2 Minimal Temporal Expressions

Minimalism of temporal logic expressions parallels the Boolean case. Temporal logic expressions are minimal, if they “do not include each other”. In the temporal logic special care is necessary, though, because of three differences compared to the Boolean case: first, their are other and additional logic operators; second, negated events have special meaning; third, properties of commutativity and associativity are restricted. Moreover, temporal expressions can be structurally non-minimal as well as temporally non-minimal, see chapters 4.3.2.1 and 4.3.2.2, respectively. First some groundwork has to be laid, though.

Minimal Temporal Failure Function

Using the temporal transformation laws from above, temporal expressions ϖ may be transformed into a TDNF, which is similar to the Boolean DNF. For readability, (4.16) is repeated here:

$$\varpi = \bigvee_{j=1}^{\zeta} ES_j = ES_1 \vee ES_2 \vee \dots \vee ES_{\zeta} . \quad (4.110)$$

ζ denotes the number of event sequences ES in ϖ , which need not to be minimal at this stage.

Then, the corresponding minimal form consists of ξ *minimal cutset sequences* (MCSS), which are OR connected:

$$\varpi = \bigvee_{j=1}^{\xi} MCSS_j = MCSS_1 \vee MCSS_2 \vee \dots \vee MCSS_{\xi} , \text{ with } \xi \leq \zeta . \quad (4.111)$$

Condition of Minimality

In the temporal logic “minimal” also means, that none of the $MCSS_j$ “covers” or “includes” any other $MCSS_{j'}$ (where $j, j' \in \{1, 2, \dots, \xi\}$ and $j' \neq j$).

The sections below show that the criterion for temporal expressions being minimal is very similar to the Boolean criterion in (4.107).

Event sequences are minimal, if all pairs of $MCSS_j$ and $MCSS_{j'}$ with $j, j' \in \{1, 2, \dots, \xi\}$ and $j' \neq j$ follow

$$MCSS_{j'} \not\subseteq MCSS_j \iff MCSS_j \wedge MCSS_{j'} \neq MCSS_{j'} \quad \text{and} \quad (4.112)$$

$$MCSS_j \not\subseteq MCSS_{j'} \iff MCSS_j \wedge MCSS_{j'} \neq MCSS_j . \quad (4.113)$$

For this relation a new operator is introduced:

$$MCSS_j \not\supseteq MCSS_{j'} \quad (4.114)$$

implies that $MCSS_j$ and $MCSS_{j'}$ are minimal.

One difference to the Boolean case is that writing temporal expressions in their TDNF form usually requires the use of negated events; this comes from the temporal distributive laws, see chapter 4.2.10, and requires a discussion on minimal temporal expressions with negated events.

4.3.2.1 Structurally Non-Minimal Temporal Expressions

Temporal expressions are *structurally non-minimal*, if one of them is a special case of the other expression. Structurally non-minimal expressions may be transformed into a minimal form by applying the temporal laws of absorption (chapter 4.2.11) and the temporal law for intersections (chapter 4.2.12).

4.3.2.2 Temporally Non-Minimal Temporal Expressions

Beyond the structural aspect of non-minimality there is the question of minimality in temporal expressions like

$$(\neg B \wedge A) \vee (A \vec{\wedge} B) . \quad (4.115)$$

Checking for minimality according to (4.113) shows that these two terms are not minimal.

From

$$(\neg B \wedge A) \wedge (A \vec{\wedge} B) \quad (4.116)$$

follows with (4.53), that

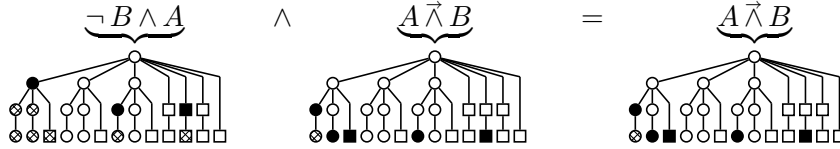
$$\begin{aligned} (\neg B \wedge A) \wedge (A \vec{\wedge} B) &= \\ &= [\neg B \wedge (A \wedge (A \vec{\wedge} B))] \vee [(A \vec{\wedge} B) \vec{\wedge} (A \vec{\wedge} B)] \vee [A \vec{\wedge} (B \vec{\wedge} (A \vec{\wedge} B))] . \end{aligned} \quad (4.117)$$

The first sub-expression on the right side is then reduced by applying (4.51), which yields

$$\neg B \wedge (A \wedge (A \vec{\wedge} B)) = \neg B \wedge (A \vec{\wedge} B) = \text{False} . \quad (4.118)$$

The second sub-expression is then also reduced to *False* by applying the temporal law of contradiction, see (4.38). Then, the remaining

$$(\neg B \wedge A) \wedge (A \vec{\wedge} B) = A \vec{\wedge} (B \vec{\wedge} (A \vec{\wedge} B)) = A \vec{\wedge} (A \vec{\wedge} B) = A \vec{\wedge} B \quad (4.119)$$



does not satisfy the minimality condition from (4.113). Therefore, (4.115) is not minimal, which is also shown by the sequential failure trees, as the sub-expression $A \vec{\wedge} B$ consists only of such expressions that are non-minimal with regard to $\neg B \vec{\wedge} A$. Thus, the minimal form is given as $\neg B \vec{\wedge} A$, which “covers” the second term $A \vec{\wedge} B$.

Generalization

The example from above may be generalized with the laws of transformation for negated events from chapter 4.2.8.3. From (4.59) follows $\neg X \wedge ES$ with $X \notin ES$ is temporally minimal to all temporal expressions with ES occurring before X , i.e. $ES \vec{\wedge} X$.

As $(\neg X \wedge ES) \vee (ES \vec{\wedge} X)$ with $X \notin ES$ is non-minimal because of the temporal sequence of the events, this effect is called *temporal non-minimality*.

Two More Examples

$(\neg B \wedge A) \vee (C \vec{\wedge} A)$ is already given in minimal form, as (4.53) and (4.113) hold:

$$\underbrace{\neg B \wedge A}_{\text{Tree 1}} \wedge \underbrace{C \vec{\wedge} A}_{\text{Tree 2}} = \underbrace{\neg B \wedge (C \vec{\wedge} A)}_{\text{Tree 3}} . \quad (4.120)$$

The sequential failure trees prove that each of the expressions includes failure nodes, which are unique to this expression and not part of the other.

However, $\neg B \wedge A$ is the minimal form of all such event sequences that include A but not $B \vec{\wedge} A$, i.e. (without SAND) $\neg B \wedge (A \vec{\wedge} C)$, $\neg C \wedge (A \vec{\wedge} B)$, $A \vec{\wedge} B \vec{\wedge} C$, $A \vec{\wedge} C \vec{\wedge} B$, and $C \vec{\wedge} A \vec{\wedge} B$. Exemplarily, this is shown with one of these expressions:

$$\begin{aligned} (\neg B \wedge A) \wedge (A \vec{\wedge} C \vec{\wedge} B) &= [\neg B \wedge (A \wedge (A \vec{\wedge} C \vec{\wedge} B))] \vee \\ &\vee [A \vec{\wedge} B \vec{\wedge} ((A \vec{\wedge} C \vec{\wedge} B) \vec{\wedge} A)] \vee \\ &\vee [A \vec{\wedge} (B \vec{\wedge} (A \vec{\wedge} C \vec{\wedge} B))] = \\ &= \text{False} \vee \text{False} \vee A \vec{\wedge} (A \vec{\wedge} C \vec{\wedge} B) = A \vec{\wedge} C \vec{\wedge} B . \end{aligned} \quad (4.121)$$

As the sequential failure trees show, (4.113) is not complied with; and $(\neg B \vec{\wedge} A) \vee (A \vec{\wedge} C \vec{\wedge} B)$ is, thus, non-minimal.

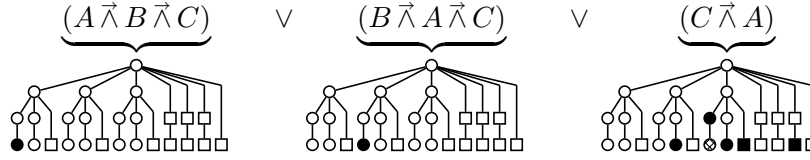
4.3.3 Disjoint Temporal Expressions

Minimal temporal expressions are not necessarily also mutually exclusive (disjoint). For example, the failure function $\varpi = (\neg B \wedge A) \vee (C \vec{\wedge} A)$ is given in minimal form. But the two event sequences $\neg B \wedge A$ and $C \vec{\wedge} A$ are not mutually exclusive; instead, $\neg B \wedge (C \vec{\wedge} A)$ is an intersection, see (4.120).

The sections below discuss mutually exclusive temporal expressions and a method for transforming them into mutually exclusive temporal expressions.

4.3.3.1 Condition for Disjointness

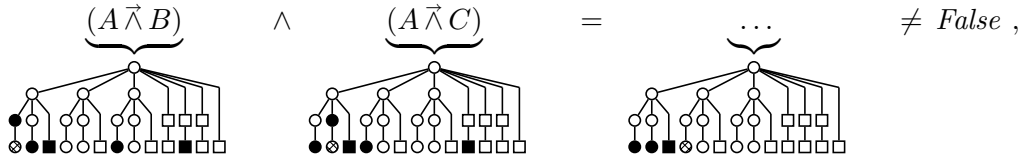
In analogy to chapter 4.3.1, two temporal expressions are mutually exclusive (disjoint), if their conjunction (AND connection) yields *False*, i.e. if there is no intersection between them. When illustrated by sequential failure trees, disjoint temporal expressions do not have any failure nodes in common. In the following example, a temporal expression has three disjoint sub-expressions:



Thereby,

$$\begin{aligned} (A \vec{\wedge} B \vec{\wedge} C) \wedge (B \vec{\wedge} A \vec{\wedge} C) &= \text{False} , \\ (A \vec{\wedge} B \vec{\wedge} C) \wedge (C \vec{\wedge} A) &= \text{False} , \\ (B \vec{\wedge} A \vec{\wedge} C) \wedge (C \vec{\wedge} A) &= \text{False} . \end{aligned}$$

On the other hand, there are intersections in the following example:



as

$$\begin{aligned} (A \vec{\wedge} B) \wedge (A \vec{\wedge} C) &= [(A \vec{\wedge} B) \vec{\wedge} (A \vec{\wedge} C)] \vee [(A \vec{\wedge} B) \vec{\wedge} (A \vec{\wedge} C)] \vee [(A \vec{\wedge} C) \vec{\wedge} (A \vec{\wedge} B)] = \\ &= [A \vec{\wedge} B \vec{\wedge} C] \vee [A \vec{\wedge} (B \vec{\wedge} C)] \vee [A \vec{\wedge} C \vec{\wedge} B] . \end{aligned}$$

4.3.3.2 Structurally and Temporally Disjoint Temporal Expressions

In the TFTA's temporal logic there are two types of disjointness:

1. An event can not be *True* and *False* at the same time. Therefore and in analogy to the Boolean logic, two expressions are disjoint, if one of them includes a non-negated event and the other expression includes the negation of the same event. For instance, $\neg A \wedge B$ and $A \vec{\wedge} B$ are mutually exclusive (disjoint). In general, this type of disjointness is expressed in (4.51) and (4.52).

2. Other than Boolean expressions, temporal expressions can be mutually exclusive because of the possibility of temporal contradictions. Following from the temporal laws of completion and the temporal law of contradiction (see chapter 4.2.2 and 4.2.3, respectively), two temporal expressions are disjoint, if the same events are included in both, but in different sequences. Therefore, $B \vec{\wedge} A$ und $A \vec{\wedge} B$ are, e.g., disjoint without any negated events.

In both cases the lack of any intersections indicates that the expressions are mutually exclusive. Therefore, the condition for disjointness from chapter 4.3.3.1 is applicable for temporal as well as Boolean expressions, see (4.109). And in consequence, temporal and Boolean expressions do not differ significantly regarding being mutually exclusive.

4.3.3.3 Disjoint Separation Using Temporal Minterms

Temporal minterms are event sequences, which consists of all u parameters of a temporal logic function of size u , and each parameter is included exactly once.

Temporal minterms are used in order to split a temporal expression into disjoint event sequences. In this form they are especially well suited for later probabilistic quantification. See chapter 4.3.1 for further background.

These expressions may be deduced using a method which is similar to Shannon's segmentation for Boolean expressions:

1. The relevant temporal function ϖ with u different parameters has to be given as TDNF. If not, ϖ is transformed into a TDNF using the temporal logic laws from above.
2. The first event sequence is chosen: $ES = ES_1$.
3. If ES consists of all u parameters, goto step seven.
4. Choose the first parameter X which is missing in ES .
5. ES is then transformed into its disjoint form by using

$$ES \implies ES \wedge (\neg X \vee X) = (\neg X \wedge ES) \vee (X \wedge ES) \quad (4.122)$$

6. Repeat step five for each of the other parameters that are missing in ES .
7. If the chosen ES is not the last event sequence in ϖ , choose the next event sequence ES and goto step three.
8. Check whether the resulting expressions are minimal by applying the transformation laws of the temporal logic and specifically the temporal laws of absorption.

This method and workflow are shown on two examples in appendix A.3, see page 120.

4.4 Simplification Using Extended Event Sequences and Extended TDNF and Extended MCSS

Chapter 4.1.5.1 discussed "normal" temporal expressions and the temporal logic, which allows to transform temporal expressions ϖ into their – possibly minimal and mutually exclusive (disjoint) – TDNF. The TDNF describes all the event sequences that lead to the occurrence of the TOP event; it is well suited for further qualitative cutset analyses, and it provides the basis for probabilistic quantification of the failure function.

4.4.1 Motivation and Requirements

Although both of the TFTA's goals from chapter 3.2 are met with these “normal” temporal expressions, their practical useability is limited because of the high number of resulting event sequences. For instance, the relatively simple temporal expression $A \bar{\wedge} (B \wedge C) \bar{\wedge} (D \wedge E)$ already provides 32 different temporal minterms (chapter 5.5.2) – and that is without even taking SANDs into account. This combinatorial blow-up of the number of event sequences mainly stems from applying the temporal law of completion (see chapter 4.2.2).

On the one hand, transformations according to the temporal logic are necessary for transforming complex expressions into manageable ones. On the other hand, clarity and readability of the results depend very much on the (low) number of such sub-expressions.

It is, therefore, sensible to simplify a complex temporal expression only so far, as to obtain useable, and especially minimal, sub-expressions, while at the same time keep the number of such sub-expressions as small as possible.

Thus, there are certain requirements on such a simplified temporal form:

1. The simplified form shall also allow qualitative as well as probabilistic analyses.
2. The simplified form shall also be able to provide temporal expressions in a normal form.
3. Each of the event sequences of this normal form shall be minimal.
4. Each of the event sequences of this normal form shall be directly quantifiable.
5. For probabilistic quantification, the event sequences shall be mutually exclusive.

The extended TDNF, as introduced in chapter 4.1.5.2, is one possibility to meet this requirements.

In extended TDNF temporal expressions consist of normal (atomic and non-atomic) core events as well as *extended core events*, such as

$$eK = X_1 \wedge X_2 \wedge \dots \quad (4.123)$$

Event sequences with extended core events are called *extended event sequences*, see the grammar of temporal logics in chapter 4.1.5.

Using this form is useful, if all sequences of specific events contribute equally to the TOP event. The extended form combines these “real” events and reduces modelling effort, and allows concise presentation of temporal expressions.

Without the extended form, temporal expressions are transformed in order to generate their TDNF consisting of event sequences only, which themselves consist of core events. Each core event stands for events which occur at a specific, though relative, point in time. An expression $A \bar{\wedge} (B \bar{\wedge} C)$, for example, indicates, that an atomic core event A occurred before later both events B and C happened simultaneously. The event sequences indicates clearly, which event occurs when.

Now, with the extended form, temporal expressions are transformed in order to generate their extended TDNF. The latter includes both, normal event sequences, consisting of normal core events, and extended event sequences, consisting of normal and extended core events.

Extended core events indicate, that at a given point in time certain events *have happend*. An expression $A \bar{\wedge} (B \wedge C)$, for example, indicates, that an atomic event core event A has occurred before later events B and C have occurred. No statement is made on the real times at which the events B and C occurred that form the extended core event. The extended form neither

defines nor restricts the sequence between B and C ; it solely describes a “latest possible” time for occurrence.

Extended event sequences may contain more than one extended event sequence, as e.g. in $(A \wedge B) \vec{\wedge} (C \wedge D)$. If events are included within the same extended event sequence more than once, then they need further transformation/simplification.

On the other hand, it disagrees with the extended TDNF to combine several (extended) event sequences with an AND. Instead, further transformation/simplification is necessary first. For example, only the simplification of $(A \vec{\wedge} B) \wedge C$ according to the laws of temporal logic provides a correct extended TDNF:

$$(A \vec{\wedge} B) \wedge C = [(A \wedge C) \vec{\wedge} B] \vee [A \vec{\wedge} (B \bar{\wedge} C)] \vee [A \vec{\wedge} B \vec{\wedge} C] . \quad (4.124)$$

4.4.2 Using Extended Temporal Expressions

The decision for using the extended form is taken during qualitative transformation of the temporal failure function:

- The Boolean distributive law gets priority over the temporal law of completion.
- AND connections are not broken up, if the AND connected events
 - are event sequences without negated events and
 - are pairwise coprime as well as coprime to the rest of the (extended) event sequence which is currently looked at.

In general, the temporal logic rules from chapter 4.2 and 4.3 apply to extended core events and extended event sequences, too. Extended core events are handled as entities, i.e. they are handled in analogy to normal non-atomic core events like $X_1 \bar{\wedge} X_2 \bar{\wedge} \dots$

There are additional transformation laws specifically for the extended form. These laws are discussed in the following sections.

Laws of Contradiction for Extended Event Sequences

The law of contradiction for normal temporal expressions (chapter 4.2.3) does not directly apply to extended event sequences. An example: the expression $(A \wedge B) \vec{\wedge} (B \wedge C)$ consists of two extended core events, which both include the same basic event B . This does not yield *False*, though. Instead, it may be further transformed using (4.48), which yields

$$(A \wedge B) \vec{\wedge} (B \wedge C) = (A \wedge B \wedge B) \vec{\wedge} C = (A \wedge B) \vec{\wedge} C . \quad (4.125)$$

On the other hand, extended event sequences may, of course, result in contradictions. The following three cases differ from each other, and together they form the *law of contradiction for extended event sequences*:

First and in analogy to (4.39), for extended event sequences with normal and extended core events eK there is

$$eK_1 \vec{\wedge} eK_2 \vec{\wedge} \dots \vec{\wedge} eK_n = \text{False} , \quad (4.126)$$

if $\exists eK_i = eK_j$ for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. This may be shown by transformation of the extended form using (4.37) and (4.77). For example,

$$(A \wedge B) \vec{\wedge} (A \wedge B) = (A \wedge B) \vec{\wedge} [(A \vec{\wedge} B) \vee (B \vec{\wedge} A) \vee (A \bar{\wedge} B)] =$$

$$\begin{aligned}
&= [(A \wedge B \wedge A) \vec{\wedge} B] \vee [(A \wedge B \wedge B) \vec{\wedge} A] \vee [(A \wedge B) \vec{\wedge} (A \vec{\wedge} B)] = \\
&= \textit{False} .
\end{aligned} \tag{4.127}$$

Second, an extended event sequences yields *False* because of a contradiction if it has an extended core event eK together with a normal core event K , which must occur *later* in the event sequence, and if there is at least one event X which apperas in K as well as in eK :

$$eK_1 \vec{\wedge} eK_2 \vec{\wedge} \dots \vec{\wedge} K_j \vec{\wedge} \dots = \textit{False} , \tag{4.128}$$

if $\exists (X \in eK_i) \wedge (X \in K_j)$ for $i < j$. K may be an atomic or non-atomic core event. For example, expression $(A \wedge B) \vec{\wedge} B$ yields a contradiction, as it requires that A as well as B have occurred before B occurs. The expression $(A \wedge B) \vec{\wedge} (A \vec{\wedge} C)$ also yields a contradiction, as it requires that A as well as B have occurred before A and C occur simultaneously. In both cases, though, there is no contradiction, if the normal core event occurs *before* the extended core event: For instance, $A \vec{\wedge} (A \wedge B) = A \vec{\wedge} B$ and $(A \vec{\wedge} C) \vec{\wedge} (A \wedge B) = (A \vec{\wedge} C) \vec{\wedge} B$.

Third, an extended event sequences yields *False* because of a contradiction if it contains more than one normal core event, and the normal law of contradiction from (4.42) applies to these core events.

Using Negated Events in Extended Event Sequences and Extended Core Events

Handling of negated events is also quite similar to the discussions from chapter 4.2.8. But there are certain additions for extended event sequences and extended core events.

Negation of extended event sequences is the same as in (4.65), but extended core events are treated as entities.

Extended core events are negated by using de Morgan's theoremes:

$$\neg eK = \neg(X_1 \wedge X_2 \wedge \dots) = \neg X_1 \vee \neg X_2 \vee \dots . \tag{4.129}$$

Negated extended core events are negated events, and as such are included into (extended) event sequences with negated events; see chapter 4.2.8 for details. Additionally to (4.51) and (4.52),

$$\neg A \wedge (\dots \vec{\wedge} (A \wedge \dots) \vec{\wedge} \dots) = \textit{False} . \tag{4.130}$$

Temporal Laws for Intersections of Extended Event Sequences and Extended Core Events

There is a special law for intersections of extended event sequences and extended core events, which provides

$$A \vec{\wedge} (A \wedge B \wedge \dots) = A \vec{\wedge} B \wedge \dots . \tag{4.131}$$

Its correctness is easily demonstrated by breaking up the extended core event.

4.5 Summary

The TFTA's temporal logic described in this chapter extends the conventional Boolean FTA for non-repairable components/failures; it allows to model and analyze event sequences.

The TFTA is an extension to Boolean algebra and logic and does not rely on state-based modelling techniques. Apart from Boolean operators for the conventional conjunction, disjunction, and negation, the TFTA has two additional operators PAND and SAND; these are "specialized conjunctions" which differentiate between event sequences and simultaneous events.

Using conventional Boolean logic transformations and additional laws of transformation for temporal expressions, it is possible to transform complex temporal expressions into a temporal disjunctive normal form (TDNF). The TDNF consists of separated event sequences. The latter may be reduced into their minimal form, so called MCSS. The TFTA thus allows efficient and meaningful qualitative analyses, just as the conventional FTA does.

As an extension to the Boolean algebra, the TFTA's temporal logic is universally applicable and not at all restricted to certain failure rate distributions.

In another step MCSS may be transformed into mutually exclusive expressions. The latter are especially well suited for direct probabilistic quantification and thus allow probabilistic analyses of temporal expressions, see the next chapter 5.

The TFTA follows the conventional FTA in notation, expressions, workflow-steps, and work products. When compared to state based dynamic methods, the TFTA, therefore, has similar positive characteristics: its logic expressions and results are similarly intuitive in use, similarly readable and comprehensible, and it has good scalability.

Simplification of temporal expressions into a minimal form (and if necessary: mutually exclusive, disjoint form, too) requires heavy effort, which is an additional cost when compared to Boolean FTA. This, on the other hand is no problem specific to the TFTA, and instead is, in principle, the same for all dynamic models.

The TFTA allows for an efficient reduction of effort, though, by means of an "extended logic form". If several sequences may be combined into a normal, i.e. Boolean, conjunction, then the extended form does not explicitly break them down. This alone highly improves the calculatory effort, which otherwise grows exponentially.

5 Probabilistic Quantification of the TFTA Method

Probable impossibilities are to be preferred to improbable possibilities.

(Aristoteles)

The quantification of the TFTA method extends the qualitative analysis. Allocation of failure rates and probabilities to basic events allows the calculation of the TOP event's failure parameters. These are then used in order to assess system characteristics like its safety integrity or expected reliability.

On the one hand, additional effort is necessary for the probabilistic quantification of the TOP event's parameters with consideration of event sequences. On the other hand, the TFTA's quantification yields smaller values than the conventional Boolean FTA.

This chapter is structured in four sections:

- Chapter 5.1 starts with the basics of probabilistic quantification of the Boolean FTA.
- Chapter 5.2 describes the concept behind the quantification of the TFTA, which is based on failure densities.
- Chapter 5.3 discusses direct quantification of the PAND and SAND operations.
- Using these, chapter 5.4 then describes the quantification of entire temporal failure functions, i.e. the calculation of the TOP event's failure probability, failure density, and failure rate.
- As these calculations require exponentially increasing calculatory effort, chapter 5.5 introduces a simplification which provides approximated failure characteristics for temporal expressions.

Note: In chapter 4 the qualitative TFTA was discussed for non-repairable components and their failures, only. This restriction also applies to the concept of quantification including chapter 5.3.1. Chapter 5.3.2 then focusses on the special case where failure parameters are distributed exponentially.

5.1 Quantification of the Boolean FTA

In the Boolean as well as the temporal FTA the probabilistic analysis of the TOP event is based on the system's TOP failure function as provided by a preceding qualitative analysis. Usually,

this logic expression is then transformed (using the transformation laws of Boolean or temporal logic) into a form, which is well suited for the task at hand (in this case: quantification).

For example, the minimal cutset form of the Boolean failure function of the system described in (4.106) is given as

$$\varphi = \bigvee_{j=1}^{\xi} MS_j = \bigvee_{j=1}^{\xi} \left(\bigwedge_{i=1}^{n_j} X_{j,i} \right).$$

This form is very clear and well suited for qualitative analysis. On the other hand, there is an equivalent but less clear form of the same failure function, as given in (4.108):

$$\varphi = \bigvee_{j=1}^{\xi} \left(MS_j \cdot \bigwedge_{i=1}^{j-1} \neg (MS_i) \right).$$

Here, the minimal cutsets are mutually exclusive (disjoint), which is less easily readable but simplifies probabilistic analyses.

The quantification of minimal cutsets of the conventional FTA, with Boolean AND and OR and NOT, is well known; it is mentioned here only for completeness.

Assuming n mutually independent events, there are

$$F_{\text{AND}}(t) = \prod_{i=1}^n F_i(t), \quad (5.1)$$

$$F_{\text{OR}}(t) = 1 - \prod_{i=1}^n (1 - F_i(t)), \quad (5.2)$$

$$f_{\text{AND}}(t) = \frac{d}{dt} F_{\text{AND}}(t) = \sum_{i=1}^n \left(f_i(t) \cdot \prod_{j=1; j \neq i}^n F_j(t) \right), \quad (5.3)$$

$$f_{\text{OR}}(t) = \frac{d}{dt} F_{\text{OR}}(t) = \sum_{i=1}^n \left(f_i(t) \cdot \prod_{j=1; j \neq i}^n (1 - F_j(t)) \right). \quad (5.4)$$

Failure functions of fault trees are usually complex expressions with non-independent events and sub-expressions. It is, thus, convenient to reduce such failure functions into their minimal cutset form before quantification, as well as to further transform the minimal cutsets into a mutually exclusive (disjoint) form. This is, for example, described in [80, 81] (and for non-monotonous functions in [82, 83]). Disjoint events simplify quantification; instead of the generic (5.2) and (5.4), the much more simple

$$F_{\text{OR}}(t) = \sum_{i=1}^n F_i(t), \quad (5.5)$$

$$f_{\text{OR}}(t) = \sum_{i=1}^n f_i(t) \quad (5.6)$$

may be used.

In monotonous fault trees with non-repairable failure events, negated events are used exclusively as conditional events; and as such, there is no failure density of negated events. This is

also true in case of TFTA, as shown by the discussions in chapter 4.2.8: negated events occur only prior to other (non-negated) events.

The probability of occurrence of a negated event $\neg X_i$ is then given by

$$F_{\neg X_i}(t) = 1 - F_{X_i}(t) = R_{X_i}(t) . \quad (5.7)$$

5.2 Quantification of the TFTA: Temporal Concept and Failure Frequencies

Other than the Boolean FTA, the temporal logic of the TFTA permits restrictions on the sequence of event occurrence in conjunctions. Any quantification of the TFTA, therefore, must also take only specific event sequences into account. This chapter explains in general, how this may be accomplished. Chapter 5.3 then uses these basics and derives specific rules for the quantification of the temporal operators PAND and SAND, respectively.

In general, failure probabilities, failure densities, and failure rates are given as [14]

$$f_X(t) = \frac{d}{dt}F_X(t) \quad \text{and} \quad (5.8)$$

$$f_X(t) = \lambda_X(t) \cdot (1 - F_X(t)) = \lambda_X(t) \cdot R_X(t) . \quad (5.9)$$

In case of constant failure rates the failure probabilities and failure densities are then given as

$$F_X(t) = 1 - e^{-\lambda_X t} \quad \text{and} \quad f_X(t) = \lambda_X \cdot e^{-\lambda_X t} . \quad (5.10)$$

5.2.1 Sequences with Two Events

In a conjunction with independent inputs (basic events) A and B there is

$$F_{A \wedge B}(t) = F_A(t) \cdot F_B(t) . \quad (5.11)$$

This is the probability, that at time t both fault tree events A and B are *True*. This is also the probability, that the failures represented by A and B have both occurred at some time during interval $]0; t]$. It is not possible, though, to make specific statements on either the sequence of these failures, nor on the absolute point in time at which the failures occurred.

Other than the failure probability $F(t)$, the failure density $f(t)$ does consider event sequences, as

$$f_{A \wedge B}(t) = \frac{d}{dt}F_{A \wedge B}(t) = f_B(t)F_A(t) + f_A(t)F_B(t) \quad (5.12)$$

and thus, using (5.9),

$$f_{A \wedge B}(t) = F_A(t)R_B(t)\lambda_B(t) + F_B(t)R_A(t)\lambda_A(t) . \quad (5.13)$$

Equation (5.13) may be interpreted as the probability per time, that [84]

- either: A has occurred at some time in interval $]0; t]$, i.e. $F_A(t)$, and B has not occurred in interval $]0; t]$, i.e. $R_B(t)$, and B will occur in the (infinitesimally) short period $]t; t + \Delta t]$ after t , i.e. $\lambda_B(t)$;

- or: B has occurred at some time in interval $]0; t]$, i.e. $F_B(t)$, and A has not occurred in interval $]0; t]$, i.e. $R_A(t)$, and A will occur in the (infinitesimally) short period $]t; t + \Delta t]$ after t , i.e. $\lambda_A(t)$.

These two possibilities represent the two sequences “ A first, and then B ” and “ B first, and then A ”, which are mutually exclusive. Therefore, their probabilities may simply be added.

This makes it possible to quantify specific event sequences. If, for example, only the event sequence “ A first, and then B ” is relevant, then

$$f_{\text{“}A \text{ first, and then } B\text{”}}(t) = F_A(t) \cdot \frac{d}{dt} F_B(t) = f_B(t) F_A(t) = \lambda_B(t) R_B(t) F_A(t) . \quad (5.14)$$

The corresponding failure probability is given by integration over the density:

$$F_{\text{“}A \text{ first, and then } B\text{”}}(t) = \int_0^t f_{\text{“}A \text{ first, and then } B\text{”}}(\tau) \cdot d\tau = \int_0^t f_B(\tau) F_A(\tau) \cdot d\tau . \quad (5.15)$$

5.2.2 Sequences with More Than Two Events

In case of more than two events, the sequence(s) of those events must also be considered that are not the “last occurring” events. For an AND gate with three inputs A , B , and C , where event sequence “ A first, and then B , and then C ” is relevant, it is thus not sufficient to simply take the derivative of $F_{A \wedge B \wedge C}(t)$, as

$$f_{A \wedge B \wedge C}(t) = f_A(t) F_B(t) F_C(t) + f_B(t) F_A(t) F_C(t) + f_C(t) F_A(t) F_B(t) . \quad (5.16)$$

None of the expressions on the right side of (5.16) represents the relevant event sequence “ A first, and then B , and then C ”. E.g., $f_C(t) F_A(t) F_B(t)$ is the density contribution of “ A and B first, and then C ”; it thus represents both event sequences “ A first, and then B , and then C ” and “ B first, and then A , and then C ”.

On the other hand, it is possible to correctly take the “not-last-occurring” events (here: A and B) into account. It is necessary to treat “ A first, and then B ” as an entity by itself, thus

$$\begin{aligned} f_{\text{“}A \text{ first, and then } B, \text{ and then } C\text{”}}(t) &= \\ &= f_{\text{“(}A \text{ first, and then } B) \text{ first, and then } C\text{”}}(t) = f_C(t) F_{\text{“}A \text{ first, and then } B\text{”}}(t) . \end{aligned}$$

Using (5.15) the failure density is then given as

$$f_{\text{“}A \text{ first, and then } B, \text{ and then } C\text{”}}(t) = f_C(t) \int_0^t f_B(\tau) F_A(\tau) \cdot d\tau . \quad (5.17)$$

Finally, the failure probability is obtained by integration:

$$F_{\text{“}A \text{ first, and then } B, \text{ and then } C\text{”}}(t) = \int_0^t f_C(\tau) \int_0^\tau f_B(\tau') F_A(\tau') \cdot d\tau' \cdot d\tau . \quad (5.18)$$

This method allows quantification of arbitrarily complex sequences with more than two events.

5.2.3 What Parameter to Use in Probabilistic Analyses?

Safety standards, as e.g. IEC 61508 or ISO 26262, require verification that systems meet specific failure rates $\lambda(t)$ [85]; evidence to verify that may be provided using probabilistic FTA. If the failure probability $F(t)$ and failure frequency $f(t)$ are given, then the failure rate is derived from (5.9).

In most cases it is not necessary to provide the failure rate, though. In the safety domain, the absolute probabilities of failure events occurring is usually so small that $F(t) \ll 1$, and thus with (5.9)

$$f(t) \approx \lambda(t) . \quad (5.19)$$

In such cases, the failure frequency is a good approximation of the failure rate, and may be directly used as target value.

5.3 Quantification of the PAND and SAND Operators

Based on the generic method of quantification of event sequences in chapter 5.2, the TFTA's temporal operations may now be quantified.

But first it is helpful to grasp the temporal meaning of PAND and SAND operations probabilistically; this is accomplished in chapter 5.3.1. Chapter 5.3.2 compares the TFTA with a state-based model as reference, and thereby demonstrates the correctness of the TFTA's quantification.

5.3.1 Quantification Using Logic Functions

The failure probability is defined as the expectancy value for the occurrence of a failure [14], and thus

$$F_i(t) = E[X_i(t) = \text{True}] = E[X_i(t)] . \quad (5.20)$$

Accordingly, the failure frequency is defined as [59]

$$\begin{aligned} f_i(t) &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} E[(X_i(t) = \text{False}) \wedge (X_i(t + \Delta t) = \text{True})] = \\ &= \lim_{\Delta t \rightarrow 0} \frac{1}{\Delta t} E[\neg X_i(t) \wedge X_i(t + \Delta t)] . \end{aligned} \quad (5.21)$$

By simple transformation an equivalent form is provided, which is specifically helpful for the further discussion:

$$f_i(t)\Delta t + o(\Delta t) = E[\neg X_i(t) \wedge X_i(t + \Delta t)] \quad \text{where} \quad \lim_{\Delta t \rightarrow 0} \frac{o(\Delta t)}{\Delta t} = 0 . \quad (5.22)$$

PAND Operation

The PAND operator in $A \vec{\wedge} B$ describes the occurrence of B at time t after A has already occurred. Non-infinitesimally, this implies that

- at time t event A has already occurred, and event B has not yet occurred, and
- at $t + \Delta t$ both, event A as well as event B , have occurred.

Therefore,

$$A(t) \wedge \neg B(t) \wedge A(t + \Delta t) \wedge B(t + \Delta t) , \quad (5.23)$$

from which with (5.22) follows (assuming independent events A and B), that

$$\begin{aligned} f_{A \vec{\wedge} B}(t)\Delta t + o(\Delta t) &= E[A(t) \wedge \neg B(t) \wedge A(t + \Delta t) \wedge B(t + \Delta t)] = \\ &= E[A(t) \wedge A(t + \Delta t)] \cdot E[\neg B(t) \wedge B(t + \Delta t)] . \end{aligned} \quad (5.24)$$

The expectancy value $E[\neg B(t) \wedge B(t + \Delta t)]$ may be directly replaced by (5.22). The expectancy value $E[A(t) \wedge A(t + \Delta t)]$, on the other hand, is not equal to the simple product of the expectancy values of events $A(t)$ and $A(t + \Delta t)$, as they are not independent from each other. Instead,

$$E[A(t) \wedge A(t + \Delta t)] = E[A(t + \Delta t) | A(t)] \cdot E[A(t)] = E[A(t)] , \quad (5.25)$$

as a failure, that has occurred at time t , “is still occurred” at $t + \Delta t$. Thus,

$$f_{A \vec{\wedge} B}(t)\Delta t + o(\Delta t) = F_A(t) \cdot [f_B(t)\Delta t + o(\Delta t)] . \quad (5.26)$$

Division by Δt , and $\Delta t \rightarrow 0$, yields

$$f_{A \vec{\wedge} B}(t) = \lim_{\Delta t \rightarrow 0} \left(F_A(t) \cdot \left[f_B(t) + \frac{o(\Delta t)}{\Delta t} \right] - \frac{o(\Delta t)}{\Delta t} \right) , \quad (5.27)$$

and finally

$$f_{A \vec{\wedge} B}(t) = F_A(t) \cdot f_B(t) . \quad (5.28)$$

Obviously, $A \vec{\wedge} B$ from (5.28) is therefore equal to the sequence “ A first, and then B ” from (5.14). This allows to state the failure probability function of the PAND operator:

$$F_{A \vec{\wedge} B}(t) = \int_0^t F_A(\tau) f_B(\tau) \cdot d\tau . \quad (5.29)$$

SAND Operation

The SAND operator in $A \bar{\wedge} B$ describes the exact simultaneous occurrence of A and B at time t . Non-infinitesimally, this implies that

- at time t neither event A nor event B has already occurred, and
- at $t + \Delta t$ both, event A as well as event B , have occurred.

Therefore,

$$\neg A(t) \wedge \neg B(t) \wedge A(t + \Delta t) \wedge B(t + \Delta t) , \quad (5.30)$$

from which follows (assuming independent events A and B), that

$$\begin{aligned} f_{A \bar{\wedge} B}(t)\Delta t + o(\Delta t) &= E[\neg A(t) \wedge A(t + \Delta t)] \cdot E[\neg B(t) \wedge B(t + \Delta t)] = \\ &= [f_A(t)\Delta t + o(\Delta t)] \cdot [f_B(t)\Delta t + o(\Delta t)] = \\ &= f_A(t) \cdot \Delta t \cdot f_B(t)\Delta t + o(\Delta t) \cdot [\dots] . \end{aligned}$$

Division by Δt , and $\Delta t \rightarrow 0$, yields

$$f_{A\bar{A}B}(t) = \lim_{\Delta t \rightarrow 0} \left(f_A(t)f_B(t)\Delta t + \frac{o(\Delta t)}{\Delta t}[\dots] - \frac{o(\Delta t)}{\Delta t} \right), \quad (5.31)$$

and finally

$$f_{A\bar{A}B}(t) = 0. \quad (5.32)$$

This implies that the probability of exact simultaneous occurrence of two independent events is always 0; every small deviation from simultaneousness is already covered – probabilistically – by the two PAND sequences of these events. Therefore,

$$F_{A\bar{A}B}(t) = 0, \quad (5.33)$$

$$\lambda_{A\bar{A}B}(t) = 0. \quad (5.34)$$

Although the SAND operator may seem unnecessary from this probabilistic point of view, it is essential for the qualitative transformation of temporal expressions, as well as for qualitative analyses. Specifically, it provides the temporal law of idempotency in (4.43), which serves as an important filter for the simplification of temporal expressions.

5.3.2 Quantification Using Comparison with State Diagrams

Note: The statements up to (5.34) apply universally. After that, the further statements discuss exponentially distributed parameters, only.

Looking back, chapter 5.2 approaches the question of quantification of the PAND operation from the definitions of the relevant parameters. Chapter 5.3 then demonstrates, that the logical meaning of PAND and SAND operations yields identical results, respectively.

In this chapter these results are compared to a reference model in order to confirm them absolutely.

This comparison is split into two parts. First, the Boolean AND and OR operations are quantified, then the quantification is extended to the temporal PAND and SAND operations using the law of completion from chapter 4.2.2.

Boolean Operations

Figure 5.1 shows the state diagram of an example system consisting of two non-repairable components A and B which have constant transition- and failure rates $\lambda_{i,j}$; this diagram is the same as in figure 2.1.

The state probabilities $P_i(t)$ are given by the following system of differential equations:

$$\begin{bmatrix} \dot{P}_1(t) \\ \dot{P}_2(t) \\ \dot{P}_3(t) \\ \dot{P}_4(t) \end{bmatrix} = \begin{bmatrix} -(\lambda_{1,2} + \lambda_{1,3} + \lambda_{1,4}) & 0 & 0 & 0 \\ \lambda_{1,2} & -\lambda_{2,4} & 0 & 0 \\ \lambda_{1,3} & 0 & -\lambda_{3,4} & 0 \\ \lambda_{1,4} & \lambda_{2,4} & \lambda_{3,4} & 0 \end{bmatrix} \cdot \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix}. \quad (5.35)$$

Assuming markov conditions are valid, event A and B have constant failure rates, and thus

$$\lambda_A = \lambda_{1,2} = \lambda_{3,4} \quad \text{und} \quad \lambda_B = \lambda_{1,3} = \lambda_{2,4}. \quad (5.36)$$

Solving the system of differential equations (5.35) provides four state probabilities $P_1(t)$ to $P_4(t)$. Looking from a reliability and safety point of view, these probabilities may be interpreted, depending on how components A and B interact:

- In case of parallel connection (redundant components) the system fails, if both components, A and B , fail. The system's failure function is $\varphi = A \wedge B$, and thus state 4 represents the system failure. As a consequence, $F_\varphi(t) = P_4(t)$ und $R_\varphi(t) = 1 - P_4(t) = P_1(t) + P_2(t) + P_3(t)$.
- In case of series connection the system fails, if either A or B or both, A and B , fail. The system's failure function is $\varphi = A \vee B$, and thus states 2 and 3 and 4 represent the system failure. As a consequence, $F_\varphi(t) = P_2(t) + P_3(t) + P_4(t)$ und $R_\varphi(t) = 1 - P_4(t) = P_1(t)$.

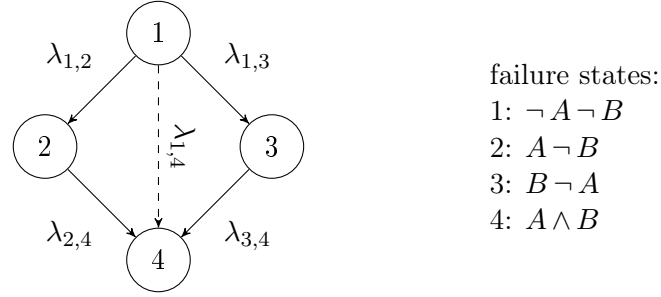


Figure 5.1: State diagram of a system consisting of two non-repairable components A and B which have constant transition- and failure rates $\lambda_{i,j}$.

Simplification

As a first step and using chapter 5.3.1, the transition representing the SAND is discarded, i.e. $\lambda_{1,4} = 0$. This is done assuming structural independence between A and B .

For the example system in (5.35) and with (5.36) this yields

$$F_{A \wedge B}(t) = P_4(t) = (1 - e^{-\lambda_A t})(1 - e^{-\lambda_B t}) = F_A(t)F_B(t) \quad (5.37)$$

and

$$F_{A \vee B}(t) = P_2(t) + P_3(t) + P_4(t) = 1 - e^{-(\lambda_A + \lambda_B)t} = 1 - [1 - F_A(t)][1 - F_B(t)] . \quad (5.38)$$

Generalization of these ideas again leads to the rules for quantification of Boolean operators, as already mentioned in (5.1) to (5.6).

PAND and SAND Operations

Temporal fault trees are quantified using their MCSS the same way as conventional fault trees are quantified using their minimal cutsets. State-transition diagrams show the correctness of the laws of completion, and they allow to derive an approach to quantification of temporal operations.

Figure 5.2 shows the example system from chapter 5.3.2 with its different event sequences. Other than figure 5.1, state 4 (“ A and B failed”) is now divided into three substates. State 4a describes the system, where A has occurred first, and then B has occurred. State 4b describes the system, where B has occurred first, and then A has occurred. State 4c describes the system, where A and B have occurred simultaneously. These state diagrams are really sequential failure trees, see page 27.

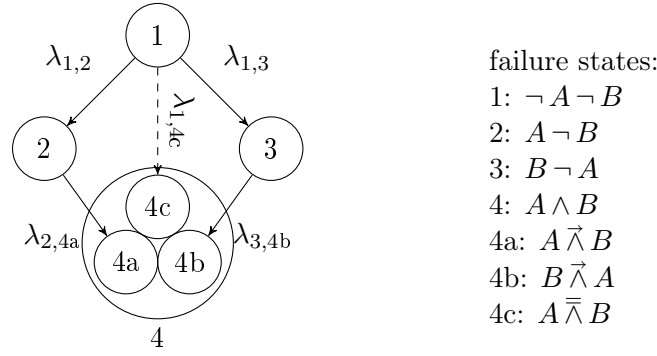


Figure 5.2: Markov model of the example system from figure 5.1 with division of state 4 “ A and B have occurred” into three substates 4a, 4b, and 4c.

These three possibilities are mutually exclusive (disjoint) and they are complete, i.e. there are no more possible ways for “ A and B have occurred”. The probability for “superstate” 4 is then given as

$$P_4(t) = P_{4a}(t) + P_{4b}(t) + P_{4c}(t) . \quad (5.39)$$

The corresponding differential equation system of the states’ probabilities may be given as the following matrix:

$$\begin{bmatrix} \dot{P}_1(t) \\ \dot{P}_2(t) \\ \dot{P}_3(t) \\ \dot{P}_4(t) \\ \dot{P}_{4a}(t) \\ \dot{P}_{4b}(t) \\ \dot{P}_{4c}(t) \end{bmatrix} = \begin{bmatrix} -(\lambda_{1,2} + \lambda_{1,3} + \lambda_{1,4c}) & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_{1,2} & -\lambda_{2,4a} & 0 & 0 & 0 & 0 & 0 \\ \lambda_{1,3} & 0 & -\lambda_{3,4b} & 0 & 0 & 0 & 0 \\ \lambda_{1,4c} & \lambda_{2,4a} & \lambda_{3,4b} & 0 & 0 & 0 & 0 \\ 0 & \lambda_{2,4a} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{3,4b} & 0 & 0 & 0 & 0 \\ \lambda_{1,4c} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \\ P_{4a}(t) \\ P_{4b}(t) \\ P_{4c}(t) \end{bmatrix} . \quad (5.40)$$

Figure 5.3 shows the markov models corresponding to the example system’s $A \wedge B$ and $A \vee B$. Assuming markovian conditions yields

$$\lambda_A = \lambda_{1,2} = \lambda_{3,4b} \quad \text{and} \quad \lambda_B = \lambda_{1,3} = \lambda_{2,4a} . \quad (5.41)$$

The solution of the set of differential equations in (5.40) for $\lambda_{1,4c} = 0$ yields the two equations known from (5.37) and (5.38):

$$F_{A \wedge B}(t) = P_4(t) = (1 - e^{-\lambda_A t})(1 - e^{-\lambda_B t}) = F_A(t)F_B(t) \quad (5.42)$$

and

$$F_{A \vee B}(t) = P_2(t) + P_3(t) + P_4(t) = 1 - e^{-(\lambda_A + \lambda_B)t} = 1 - [1 - F_A(t)][1 - F_B(t)] . \quad (5.43)$$

The law of completeness from chapter 4.2.2 allows representing an AND operation by PAND and SAND operations. Figure 5.4 shows the relevant state diagrams and failure functions.

Solving the set of differential equations in (5.40) then yields

$$F_{A \bar{\wedge} B}(t) = P_{4a}(t) = \int_0^t f_B(\tau)F_A(\tau) \cdot d\tau , \quad (5.44)$$

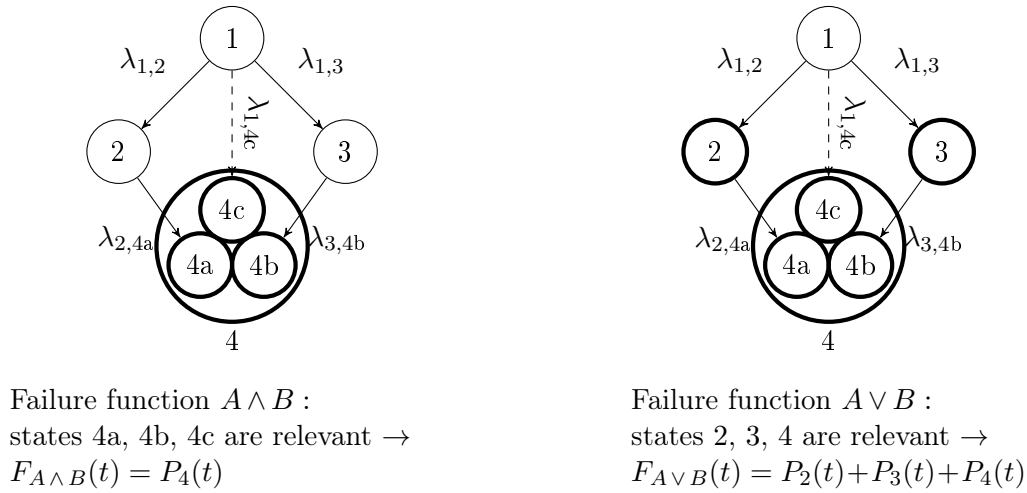


Figure 5.3: Markov model of the example system corresponding to failure functions $A \wedge B$ and $A \vee B$ and the set of differential equations in (5.40). System failure states are marked in bold.

$$F_{B \bar{\wedge} A}(t) = P_{4b}(t) = \int_0^t f_A(\tau) F_B(\tau) \cdot d\tau, \quad (5.45)$$

$$F_{A \bar{\wedge} B}(t) = P_{4c}(t) = 0. \quad (5.46)$$

Insertion of (5.44), (5.45), and (5.46) in (5.39) provides

$$\begin{aligned} F_{A \wedge B}(t) &= F_{A \bar{\wedge} B}(t) + F_{B \bar{\wedge} A}(t) + F_{A \bar{\wedge} B}(t) = \\ &= \int_0^t (f_B(\tau) F_A(\tau) + f_A(\tau) F_B(\tau)) \cdot d\tau + 0 = F_A(t) \cdot F_B(t). \end{aligned} \quad (5.47)$$

This demonstrates that the law of completion holds also from a probabilistic point of view, and shows the correctness of the calculations in chapter 5.3.

The corresponding failure frequencies and failure rates are then given by (5.8) and (5.9), respectively.

5.4 Quantification of the Temporal Failure Function

Chapter 5.2 shows the basic concept of quantifying event sequences. Applying this concept to arbitrary temporal expressions in TDNF allows the quantification of temporal fault trees, i.e. calculation of their events' – and especially their TOP event's – failure probabilities and failure rates.

5.4.1 Quantification of Event Sequences and MCSS

The probabilistic quantification of a fault tree requires, firstly, to determine its MCSS, i.e. all the critical event combinations (including their sequences) in minimal form. This is done using the rules for qualitative transformations from chapter 4.2 and 4.3. In a next step, the probabilistic

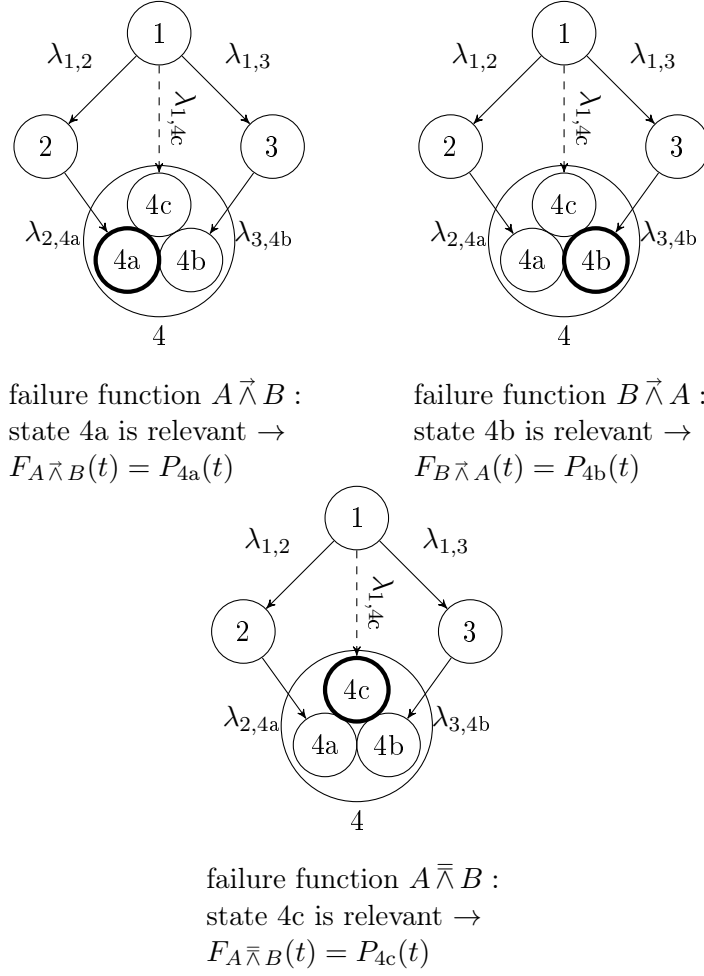


Figure 5.4: Markov model of the example system corresponding to failure functions $A \vec{\wedge} B$, $B \vec{\wedge} A$, and $A \vec{\wedge} B$ and the set of differential equations in (5.40). System failure states are marked in bold.

parameters are determined for each of the MCSS; these parameters are then used to calculate the TOP event's parameters.

Simplification for Independent Failure Events

In case of independent failure events an essential simplification is possible: According to chapter 5.3 all MCSS may be omitted that include at least one SAND. They are omitted after transforming the temporal expression into its MCSS but before the MCSS are quantified. MCSS including SANDs are only relevant for the qualitative analysis and provide no probabilistic contribution to the failure rates, failure frequencies, and failure probabilities of the temporal failure function. Only MCSS without SAND are then quantified. Thus, the quantification is carried out for MCSS of the following type:

$$X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n , \quad (5.48)$$

possibly also in conjunction with negated events

$$(\neg X_I \wedge \neg X_{II} \dots) \wedge (X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n) . \quad (5.49)$$

MCSS according to (5.48) may be directly quantified using convolutions of the failure frequencies, see (5.28) and (5.29), thus

$$\begin{aligned} f_{MCSS}(t) &= f_{X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n}(t) = \\ &= f_{X_n}(t) \int_0^t f_{X_{n-1}}(\tau^{\{1\}}) \int_0^{\tau^{\{1\}}} f_{X_{n-2}}(\tau^{\{2\}}) \dots \int_0^{\tau^{\{n-2\}}} f_{X_2}(\tau^{\{n-1\}}) \int_0^{\tau^{\{n-1\}}} f_{X_1}(\tau^{\{n\}}) \cdot \\ &\quad \cdot d\tau^{\{n\}} \cdot d\tau^{\{n-1\}} \dots d\tau^{\{2\}} \cdot d\tau^{\{1\}} . \end{aligned} \quad (5.50)$$

MCSS with Negated Events

The probabilities of negated events, which are part of MCSS, are multiplied to these results. Therefore, the MCSS' quantification according to (5.49) is given as

$$\begin{aligned} f_{MCSS}(t) &= f_{(\neg X_I \wedge \neg X_{II} \dots) \wedge (X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n)}(t) = \\ &= f_{X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n}(t) \cdot R_{X_I}(t) \cdot R_{X_{II}}(t) \dots , \end{aligned} \quad (5.51)$$

where $f_{X_1 \vec{\wedge} X_2 \vec{\wedge} \dots \vec{\wedge} X_n}(t)$ comes from (5.50).

Failure Probability and Failure Rate

The failure probabilities and failure rates corresponding to (5.50) and (5.51) are then given by using the generic equations (5.8) and (5.9).

5.4.2 Quantification of Extended Event Sequences

Extended event sequences and extended MCSS include at least one extended core event. They are, therefore, a mixture of a Boolean and a temporal logic expression. In their logical statement extended MCSS combine several real MCSS and thus cover several event sequences, see chapter 4.4.

All extended MCSS may be omitted that include at least one SAND connection; for independent events, these do not contribute probabilistically to the event probabilities.

Extended MCSS with One Extended Core Event

Let

$$\begin{aligned} X_1 \vec{\wedge} \dots \vec{\wedge} X_{k-1} \vec{\wedge} X_k \vec{\wedge} X_{k+1} \vec{\wedge} \dots \vec{\wedge} X_{n-1} \vec{\wedge} X_n \quad \text{with} \\ X_k = X_{k,1} \wedge X_{k,2} \wedge \dots \wedge X_{k,r} \end{aligned} \quad (5.52)$$

be an extended MCSS with one extended core event ($w = 1$) at position k within the PAND chain, and let the extended core event consist of r basic events that are AND connected.

Using (5.3), the failure frequency for $X_k(t)$ is then given as

$$f_{X_k}(t) = \sum_{i=1}^r \left(f_{k,i}(t) \cdot \prod_{j=1; j \neq i}^r (F_{k,j}(t)) \right) . \quad (5.53)$$

Event sequences (and thus MCSS, too) must not include the same basic event more than once, as stated by the laws of contradiction in (4.42) for normal and (4.126) for extended temporal expressions.

All events in an (extended) MCSS are thus mutually independent; therefore, the failure frequency of an extended core event may be calculated independently from the rest of the expression and using (5.53). It is then inserted into the overall failure frequency of the extended MCSS:

$$\begin{aligned}
 f_{MCSS}(t) &= f_{X_1} \bar{\wedge} \dots \bar{\wedge} X_{k-1} \bar{\wedge} X_k \bar{\wedge} X_{k+1} \bar{\wedge} \dots \bar{\wedge} X_{n-1} \bar{\wedge} X_n(t) = \\
 &= f_{X_n}(t) \int_0^t f_{X_{n-1}}(\tau^{\{1\}}) \dots \int_0^{\tau^{\{n-(k+1)\}}} f_{X_{k+1}}(\tau^{\{n-k\}}) \cdot \\
 &\quad \cdot \int_0^{\tau^{\{n-k\}}} \underbrace{f_{X_k}(\tau^{\{n-(k-1)\}})}_{\text{from (5.53)}} \int_0^{\tau^{\{n-(k-1)\}}} f_{X_{k-1}}(\tau^{\{n-(k-2)\}}) \dots \int_0^{\tau^{\{n-1\}}} f_{X_1}(\tau^{\{n\}}) \cdot \\
 &\quad \cdot d\tau^{\{n\}} \cdot d\tau^{\{n-1\}} \dots d\tau^{\{n-(k-1)\}} \dots d\tau^{\{n-(k+1)\}} \dots d\tau^{\{1\}} .
 \end{aligned} \tag{5.54}$$

Extended MCSS with Several Extended Core Events

In case of extended MCSS with more than one extended core event, thus $w > 1$,

1. the $f_{X_k}(t)$ are calculated according to (5.53) for each $k \in \{1, 2, \dots, w\}$, and
2. the resulting w failure frequencies are then inserted into the overall failure frequency of the extended MCSS; this is the same as in case of $w = 1$ from (5.54).

MCSS with Negated Events

The probabilities of negated events that are part of extended MCSS are considered in analogy to (5.51).

Failure Probability and Failure Rate

An extended MCSS' failure probability is given using (5.8) by integrating over (5.54); the corresponding failure rate is then given by (5.9).

5.4.3 Quantification of the Temporal Failure Function on TOP Level

MCSS resulting from the method in chapter 4.3 are mutually exclusive (disjoint).

Therefore, the TOP event's failure probability and failure frequency is given by (5.5) and (5.6) and is the simple sum of the probabilistic contributions of the disjoint MCSS:

$$F_{TOP}(t) = \sum_{i=1}^{\xi} F_{MCSS_i}(t) , \tag{5.55}$$

$$f_{TOP}(t) = \sum_{i=1}^{\xi} f_{MCSS_i}(t) . \tag{5.56}$$

The parameters of the disjoint MCSS come

- from chapter 5.4.1 in case of normal MCSS and
- from chapter 5.4.2 in case of extended MCSS.

5.5 Reducing the Computing Time

The calculatory effort necessary for the multiple integrals in (5.50), (5.51), and (5.54) is high; this is especially true for complex temporal fault trees and their complex failure functions. This is not helpful to the TFTA's declared goal to facilitate modelling of event sequences for large and complex systems

The following chapter therefore presents an approximatory approach to the calculation of failure probabilities, failure frequencies, and MCSS in order to significantly reduce the calculatory effort. Essential prerequisites to this approximation are

- constant failure rates of all basic events, i.e. exponentially distributed failure probabilities, and
- “small enough” failure probabilities and failure rates, i.e. the “small value assumption” from (5.19) must be valid that $\lambda t \ll 1$ and thus $f(t) \approx \lambda(t)$; in a safety context this is usually a given.

5.5.1 Temporal Terms in MCSS Format

First, temporal expressions in MCSS form are discussed; they result e.g. from qualitative transformations of a TFTA according to chapter 4.3.

MCSS Without Negated Events

The failure probability and failure rate of MCSS without negated events, which include at least one SAND, is always zero according to the discussion following page 67.

Therefore, the quantification is again based on MCSS without negated events as shown in (5.48). The corresponding failure probability is given by integration over (5.50) which yields

$$\begin{aligned}
 F_{MCSS}(t) &= F_{X_1 \bar{\wedge} X_2 \bar{\wedge} \dots \bar{\wedge} X_n}(t) = \int_0^t f_{X_1 \bar{\wedge} X_2 \bar{\wedge} \dots \bar{\wedge} X_n}(\tau) \cdot d\tau = \\
 &= \int_0^t f_{X_n}(\tau) \cdot \int_0^\tau f_{X_{n-1}}(\tau^{\{1\}}) \dots \int_0^{\tau^{\{n-2\}}} f_{X_2}(\tau^{\{n-1\}}) \cdot \int_0^{\tau^{\{n-1\}}} f_{X_1}(\tau^{\{n\}}) \cdot \\
 &\quad \cdot d\tau^{\{n\}} \cdot d\tau^{\{n-1\}} \dots d\tau^{\{1\}} \cdot d\tau .
 \end{aligned} \tag{5.57}$$

With a total of n basic events that constitute an MCSS, each MCSS represents exactly one event sequence of the $n!$ possible permutations. The probability that all n events included in an MCSS have occurred at time t is given by (5.1) for the case that no event sequences are distinguished; this yields

$$F_{X_1 \wedge X_2 \wedge \dots \wedge X_n}(t) = F_{X_1}(t) \cdot F_{X_2}(t) \dots F_{X_n}(t) = \prod_{i=1}^n F_{X_i}(t) . \tag{5.58}$$

For exponentially distributed and very small failure rates equation (5.19) then allows the approximation that

$$f(t) \approx \lambda(t) = \lambda \quad \text{and therefore} \tag{5.59}$$

$$F(t) \approx \lambda \cdot t \quad \text{for } \lambda \cdot t \ll 1 . \tag{5.60}$$

Then,

$$F_{X_1 \wedge X_2 \wedge \dots \wedge X_n}(t) \approx \lambda_{X_1} t \cdot \lambda_{X_2} t \cdots \lambda_{X_n} t = \prod_{i=1}^n (\lambda_{X_i} t) . \quad (5.61)$$

If all n failure rates $\lambda_X = \lambda_{X_1} = \dots = \lambda_{X_n}$ are equal, all $n!$ possible permutations of the event sequences occur with the same probability; thus, for each MCSS

$$F_{MCSS}(t) = \frac{1}{n!} \prod_{i=1}^n F_{X_i}(t) \approx \frac{1}{n!} \prod_{i=1}^n (\lambda_{X_i} t) . \quad (5.62)$$

Equation (5.62) is also a generic approximation in case of different failure rates, if the highest of the n failure rates satisfies the condition that

$$\max(\lambda_{X_1}; \lambda_{X_2}; \dots; \lambda_{X_n}) \cdot t \ll 1 . \quad (5.63)$$

Thus,

$$F_{MCSS}(t) \approx \frac{1}{n!} \prod_{i=1}^n (\lambda_{X_i} t) . \quad (5.64)$$

The approximation for an MCSS' failure frequency is provided accordingly. Let, without restriction to the general case, be X_n the last occurring event in a MCSS with n involved events; then

$$f_{MCSS}(t) = f_{X_1 \bar{\wedge} X_2 \bar{\wedge} \dots \bar{\wedge} X_n}(t) = f_{X_n}(t) \cdot F_{X_1 \bar{\wedge} X_2 \bar{\wedge} \dots \bar{\wedge} X_{n-1}}(t) ; \quad (5.65)$$

from this follows with (5.59) and (5.60) that

$$f_{MCSS}(t) \approx \frac{1}{(n-1)!} \cdot \lambda_{X_n} \cdot \prod_{i=1}^{n-1} (\lambda_{X_i} t) . \quad (5.66)$$

MCSS with Negated Events

An approximation for MCSS with negated events combines the procedure of chapter 5.4.1 – with the probability of negated events from (5.7) – and the quantification approach to MCSS without negated events, as discussed above. Using (5.64) and (5.66) this yields

$$\begin{aligned} F_{MCSS}(t) &= F_{(\neg X_I \wedge \neg X_{II} \dots) \wedge (X_1 \bar{\wedge} X_2 \bar{\wedge} \dots \bar{\wedge} X_n)}(t) \cdots \approx \\ &\approx \frac{1}{n!} \cdot \prod_{i=1}^n (\lambda_{X_i} t) \cdot R_{X_I}(t) \cdot R_{X_{II}}(t) \cdots , \end{aligned} \quad (5.67)$$

$$\begin{aligned} f_{MCSS}(t) &= f_{X_1 \bar{\wedge} X_2 \bar{\wedge} \dots \bar{\wedge} X_n}(t) \cdot R_{X_I}(t) \cdot R_{X_{II}}(t) \cdots \approx \\ &\approx \frac{1}{(n-1)!} \cdot \lambda_{X_n} \cdot \prod_{i=1}^{n-1} (\lambda_{X_i} t) \cdot R_{X_I}(t) \cdot R_{X_{II}}(t) \cdots . \end{aligned} \quad (5.68)$$

5.5.2 Temporal Terms in an Extended MCSS Format

The assumptions from chapter 5.5.1 still hold; specifically, no SAND connections are considered, as they do not contribute probabilistically.

Extending the method with minimized computational effort to extended MCSS requires discussing how many normal MCSS are covered by an extended MCSS.

In a very simple example, the extended MCSS $(X_1 \wedge X_2) \vec{\wedge} X_3$ covers two normal MCSS, $X_1 \vec{\wedge} X_2 \vec{\wedge} X_3$ and $X_2 \vec{\wedge} X_1 \vec{\wedge} X_3$, which are disjoint. Using (5.62), each of these two normal MCSS has a probability of

$$F_{X_1 \vec{\wedge} X_2 \vec{\wedge} X_3}(t) = F_{X_2 \vec{\wedge} X_1 \vec{\wedge} X_3}(t) \approx \frac{1}{6} F_{X_1} F_{X_2} F_{X_3} . \quad (5.69)$$

Accordingly,

$$F_{(X_1 \wedge X_2) \vec{\wedge} X_3}(t) \approx 2 \cdot \frac{1}{6} F_{X_1} F_{X_2} F_{X_3} = \frac{1}{3} F_{X_1} F_{X_2} F_{X_3} . \quad (5.70)$$

All normal MCSS that are covered by an extended MCSS are mutually exclusive (disjoint) because of the temporal law of completion. An extended MCSS' failure probability and failure frequency is therefore given as simple sum of the failure probabilities and failure frequencies of the normal MCSS that are covered by the extended MCSS.

In general and without SAND connections, the number \mathcal{T} of normal MCSS that are covered by an extended MCSS depends

- on w , which is the number of extended core events in the extended MCSS, and
- on r_i for each extended core event $i \in \{1, \dots, w\}$, which is the number of its AND connected basic events, and
- on k_i , which is the corresponding extended core event's position in the MCSS.

Some examples:

$$\begin{aligned} (X_1 \wedge X_2) \vec{\wedge} X_3 &\rightarrow w = 1 ; r = 2 ; k = 1 , \\ X_1 \vec{\wedge} (X_2 \wedge X_3) &\rightarrow w = 1 ; r = 2 ; k = 2 , \\ (X_1 \wedge X_2) \vec{\wedge} (X_3 \wedge X_4) &\rightarrow w = 2 ; r_1 = r_2 = 2 ; k_1 = 1 ; k_2 = 3 , \\ X_1 \vec{\wedge} (X_2 \wedge X_3 \wedge X_4) &\rightarrow w = 1 ; r = 3 ; k = 2 . \end{aligned}$$

In the third example it is noteworthy, that $k_2 = 3$. The position of the $i \in \{2, \dots, w\}$ -th core event is calculated including all events; even those events in "preceding" core events are considered, i.e. events on the left side of the i -th extended core event in the MCSS. SAND connections are omitted, though:

$$(X_1 \wedge X_2) \vec{\wedge} (X_3 \wedge X_4) = \left[X_1 \vec{\wedge} X_2 \vec{\wedge} (X_3 \wedge X_4) \right] \vee \left[X_2 \vec{\wedge} X_1 \vec{\wedge} (X_3 \wedge X_4) \right] . \quad (5.71)$$

The position of the second extended core event is therefore $k_2 = 3$.

In general, each extended core event i with r_i basic events and standing at position k_i covers

$$\mathcal{R}_i = \binom{(k_i - 1) + (r_i - 1)}{(k_i - 1)} \cdot r_i! \quad (5.72)$$

normal MCSS. This follows from $r_i!$ possible permutations within the extended core event. For each permutation $(k_i - 1)$ preceding events (left of the extended core event) may then hold $(k_i - 1) + (r_i - 1)$ possible positions, as described in (4.48).

Some examples:

- $(X_1 \wedge X_2) \vec{\wedge} X_3 \rightarrow w = 1 ; r = 2 ; k = 1 \rightarrow \Upsilon = 2 :$
 $\rightarrow X_1 \vec{\wedge} X_2 \vec{\wedge} X_3 , X_2 \vec{\wedge} X_1 \vec{\wedge} X_3 .$
- $X_1 \vec{\wedge} (X_2 \wedge X_3) \rightarrow w = 1 ; r = 2 ; k = 2 \rightarrow \Upsilon = 4 :$
 $\rightarrow X_1 \vec{\wedge} X_2 \vec{\wedge} X_3 , X_1 \vec{\wedge} X_3 \vec{\wedge} X_2 , X_2 \vec{\wedge} X_1 \vec{\wedge} X_3 , X_3 \vec{\wedge} X_1 \vec{\wedge} X_2 .$
- $X_1 \vec{\wedge} X_2 \vec{\wedge} (X_3 \wedge X_4) \rightarrow w = 1 ; r = 2 ; k = 3 \rightarrow \Upsilon = 6 :$
 $\rightarrow X_1 \vec{\wedge} X_2 \vec{\wedge} X_3 \vec{\wedge} X_4 , X_1 \vec{\wedge} X_2 \vec{\wedge} X_4 \vec{\wedge} X_3 , X_1 \vec{\wedge} X_3 \vec{\wedge} X_2 \vec{\wedge} X_4 ,$
 $X_1 \vec{\wedge} X_4 \vec{\wedge} X_2 \vec{\wedge} X_3 , X_3 \vec{\wedge} X_1 \vec{\wedge} X_2 \vec{\wedge} X_4 , X_4 \vec{\wedge} X_1 \vec{\wedge} X_2 \vec{\wedge} X_3 .$

With $w > 1$ extended core events the total number of covered permutations is then given as

$$\Upsilon = \prod_{i=1}^w \Upsilon_i . \quad (5.73)$$

For example, the extended MCSS $(X_1 \wedge X_2) \vec{\wedge} (X_3 \wedge X_4)$ with $w = 2, r_1 = r_2 = 2, k_1 = 1, k_2 = 3$ covers a total of $\Upsilon = \Upsilon_1 \cdot \Upsilon_2 = 2 \cdot 6 = 12$ permutations.

$$\begin{aligned} & X_1 \vec{\wedge} X_2 \vec{\wedge} X_3 \vec{\wedge} X_4 , X_1 \vec{\wedge} X_2 \vec{\wedge} X_4 \vec{\wedge} X_3 , X_1 \vec{\wedge} X_3 \vec{\wedge} X_2 \vec{\wedge} X_4 , \\ & X_1 \vec{\wedge} X_4 \vec{\wedge} X_2 \vec{\wedge} X_3 , X_3 \vec{\wedge} X_1 \vec{\wedge} X_2 \vec{\wedge} X_4 , X_4 \vec{\wedge} X_1 \vec{\wedge} X_2 \vec{\wedge} X_3 , \\ & X_2 \vec{\wedge} X_1 \vec{\wedge} X_3 \vec{\wedge} X_4 , X_2 \vec{\wedge} X_1 \vec{\wedge} X_4 \vec{\wedge} X_3 , X_2 \vec{\wedge} X_3 \vec{\wedge} X_1 \vec{\wedge} X_4 , \\ & X_2 \vec{\wedge} X_4 \vec{\wedge} X_1 \vec{\wedge} X_3 , X_3 \vec{\wedge} X_2 \vec{\wedge} X_1 \vec{\wedge} X_4 , X_4 \vec{\wedge} X_2 \vec{\wedge} X_1 \vec{\wedge} X_3 . \end{aligned}$$

In analogy to (5.67), the failure probability of an extended MCSS is approximated as

$$F_{MCSS}(t) \approx \Upsilon \cdot \frac{1}{n!} \cdot \prod_{i=1}^n (\lambda_{X_i} t) \cdot R_{X_I}(t) \cdot R_{X_{II}}(t) \cdots . \quad (5.74)$$

In analogy to (5.68), the approximated failure frequency is then given by

$$f_{MCSS}(t) \approx \Upsilon \cdot \frac{1}{(n-1)!} \cdot \lambda_{X_n} \cdot \prod_{i=1}^{n-1} (\lambda_{X_i} t) \cdot R_{X_I}(t) \cdot R_{X_{II}}(t) \cdots . \quad (5.75)$$

Summary of Chapter 5.5 Reducing the Computing Time

For constant failure rates and “small enough” failure probabilities the probabilities and rates of occurrence of each possible permutation of the events in an MCSS do not significantly differ among each other. The calculation of $F_{MCSS}(t)$ and $f_{MCSS}(t)$ is therefore almost independent of the exact event sequence information. This is beneficial, as the quantification with exact sequence information requires calculation of multiply nested integrals (see chapter 5.3) which is very costly. On the other hand, the approximation method provided in this chapter allows an estimation of $F_{MCSS}(t)$ and $f_{MCSS}(t)$ solely based on the number of events in an MCSS and their respective failure rates, see (5.67) and (5.68). It is not necessary to explicitly take the exact sequence information into consideration. Extended MCSS may also be quantified using this approximation, as shown in (5.74) and (5.75).

6 Comparing TFTA to Other Dynamic Modelling Approaches

Much may be said on both sides.

(Henry Fielding)

In this chapter the advantages of using the TFTA method are demonstrated and discussed; in order to do so, an example system (see chapter 6.1) is modelled and analyzed

- as conventional Boolean FTA in chapter 6.2,
- as dynamic fault tree (DFT) in chapter 2.3, and
- as markov model in chapter 6.4,

and these are then compared with the new TFTA approach, see chapter 6.5. The comparison models are created and analyzed using the Isograph FaultTree+ tool [50].

6.1 An Example System

An example system from [79] is shown in figure 6.1.

System Description

The relevant system function of the system under consideration is to supply point X with power. The power supply E delivers energy via switch U and two redundant paths A and B . First, U is switched to allow energy flow via path A . In case of a fault in A , switch U will redirect the energy flow via path B in order to sustain the system function.

The following component faults are considered here:

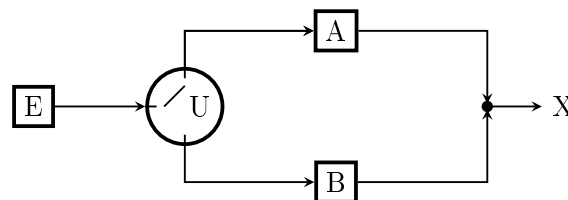


Figure 6.1: An example system used for comparing the Boolean FTA, the DFT, the markov model, and the TFTA.

E: E fails to supply energy; the corresponding failure rate is $\lambda_E = 1 \cdot 10^{-9} \frac{1}{h}$.

U: U fails to switch from A to B ; the corresponding failure rate is $\lambda_U = 5 \cdot 10^{-6} \frac{1}{h}$.

A: Internal fault of A inhibiting energy flow; the corresponding failure rate is $\lambda_A = 1 \cdot 10^{-6} \frac{1}{h}$.

B: Internal fault of B inhibiting energy flow; the corresponding failure rate is $\lambda_B = 1 \cdot 10^{-6} \frac{1}{h}$.

All components are non-repairable; all failure rates are constant; the mission time is $T_M = 400h$. The failure sequence is relevant because the failure of U before failure of A leads to a system failure, but the failure of U after switching from A , i.e. after failure of A , does not lead to a system failure. The qualitative and probabilistic results of modelling this example system using the different modelling techniques are listed in tables 6.1 and 6.2 on page 82.

6.2 Comparison with the Boolean FTA

The Boolean model is not able to take sequence information into account as relevant for this example system's failure behaviour. As an approximation to the real system diagram from figure 6.1, one of the versions from figure 6.2 must be chosen as basis for the Boolean fault tree model [79]. Figure 6.3 shows the Boolean fault trees corresponding to these two versions, which are called "Bool 1" and "Bool 2".

Qualitative and Probabilistic Calculation

The components' failure probabilities and failure frequencies at the end of the mission time are calculated using (5.10); this yields

$$F_A(T_M) = 3,9992 \cdot 10^{-4}, \quad f_A(T_M) = 9,9960 \cdot 10^{-7} \frac{1}{h}, \quad (6.1)$$

$$F_B(T_M) = 3,9992 \cdot 10^{-4}, \quad f_B(T_M) = 9,9960 \cdot 10^{-7} \frac{1}{h}, \quad (6.2)$$

$$F_U(T_M) = 1,9960 \cdot 10^{-3}, \quad f_U(T_M) = 4,9900 \cdot 10^{-6} \frac{1}{h}, \quad (6.3)$$

$$F_E(T_M) = 4,0000 \cdot 10^{-7}, \quad f_E(T_M) = 1,0000 \cdot 10^{-9} \frac{1}{h}. \quad (6.4)$$

The failure function φ is

$$\varphi_{\text{Bool 1}} = (A \vee E) \wedge (B \vee U \vee E) = [A \wedge B] \vee [A \wedge U] \vee [E] \quad \text{and} \quad (6.5)$$

$$\varphi_{\text{Bool 2}} = (A \vee U \vee E) \wedge (B \vee U \vee E) = [A \wedge B] \vee [U] \vee [E]. \quad (6.6)$$

It may be transformed into a disjunctive normal form of mutually exclusive expressions:

$$\varphi_{\text{Bool 1}} = [A \wedge B \wedge \neg E \wedge \neg U] \vee [A \wedge U \wedge \neg E] \vee [E] \quad \text{and} \quad (6.7)$$

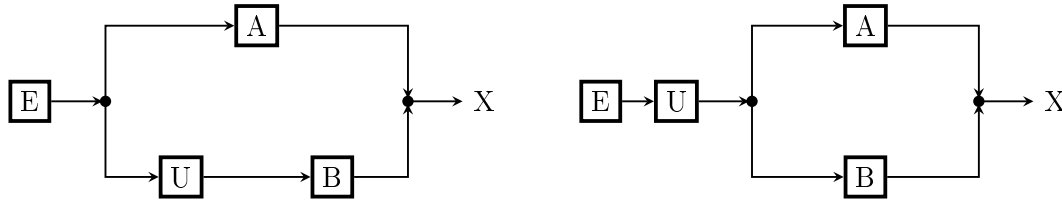


Figure 6.2: Two possible versions of Boolean approximations of the example system from figure 6.1 as basis for a conventional Boolean FTA. The left side is called "Bool 1", and the right side is called "Bool 2".

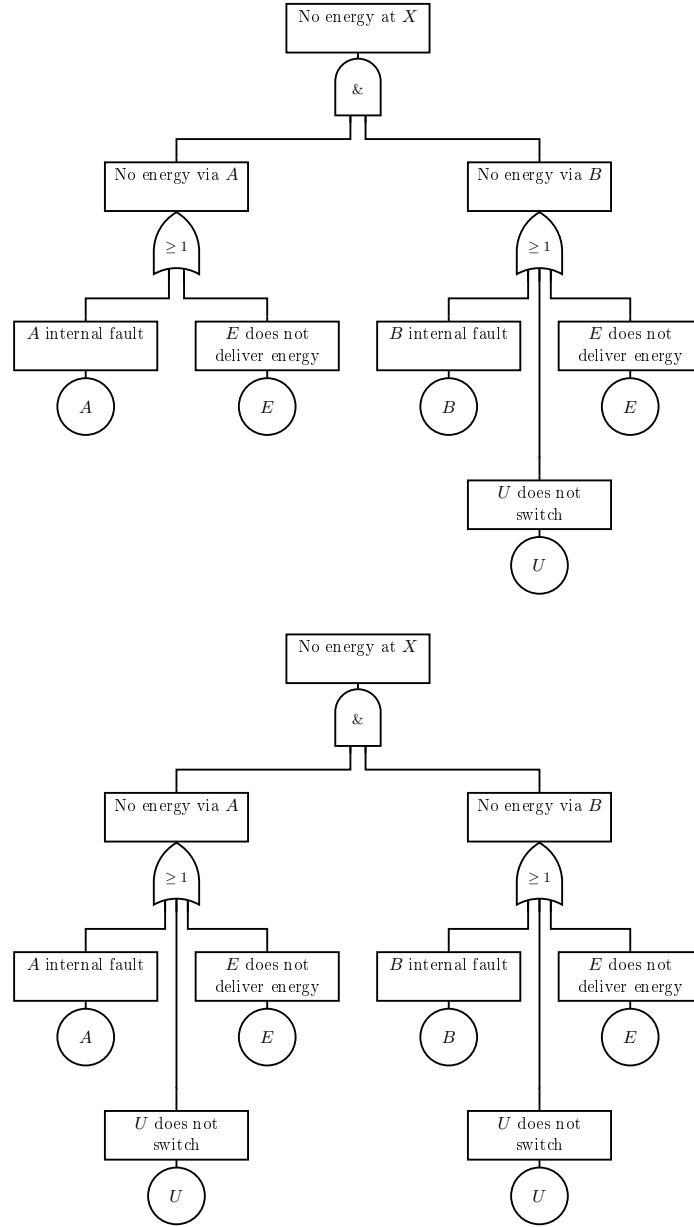


Figure 6.3: Boolean fault trees corresponding to “Bool 1” (top) and “Bool 2” (bottom).

$$\varphi_{\text{Bool 2}} = [A \wedge B \wedge \neg E \wedge \neg U] \vee [U \wedge \neg E] \vee [E] . \quad (6.8)$$

Using the failure data from above for quantification, the TOP event provides

$$F_{\text{Bool 1}}(T_M) = 1,3587 \cdot 10^{-6} , \quad f_{\text{Bool 1}}(T_M) = 5,7899 \cdot 10^{-9} \frac{1}{h} \quad \text{and} \quad (6.9)$$

$$F_{\text{Bool 2}}(T_M) = 1,9986 \cdot 10^{-3} , \quad f_{\text{Bool 2}}(T_M) = 4,9918 \cdot 10^{-6} \frac{1}{h} . \quad (6.10)$$

These results were verified using the FaultTree+ tool.

Discussion on Creating the Fault Trees

In both cases the fault tree is derived systematically from the system diagrams by following the

energy flow backwards through the system, i.e. from output X to input E . The modeller needs not think about possible event duplications, as the Boolean logic correctly eliminates those.

Discussion on Results

Qualitative analysis of the minimal cutsets shows that both cases provide system failures where no real system failure are occurring. In case of “Bool 1” the inaccuracy lies in minimal cutset $[A \wedge U]$, and in case of “Bool 2” the inaccuracy lies in minimal cutset $[U]$. Therefore, “Bool 2” is an especially conservative approximation: qualitatively, the fault tree has one additional and unnecessary single point failure; probabilistically, the fault tree yields much higher values for the TOP level failure parameters. Comparing both Boolean versions it appears clear that “Bool 1” is the more realistic model.

6.3 Comparison with Dynamic FTA (DFT Method)

Other than the Boolean model, the DFT fault tree uses PAND gates to consider event sequences, that are relevant to the system failure behaviour.

Figure 6.4 shows two versions “DFT 1” and “DFT 2” which include a dynamic module, i.e. the gate “ U fails before A ”; this module represents a markov model, see figure 2.3. For better understanding, in these figures the PAND gate is shown with its original DFT symbol from the DFT [37], i.e. an AND gate with double bars, instead of the TFTA PAND gate symbol (an AND gate with horizontal left-to-right arrow).

In “DFT 1” basic event A is meshed between the dynamic module and the Boolean part of the fault tree. Basic event A has a set sequence flag, and because of the meshing this flag is also set where A is input to the Boolean AND gate “Internal failure of A and B ”. But this sequence information is erroneous with regard to event B ; it provides probabilistically optimistic results, i.e. to small failure values.

In “DFT 2” this meshing is broken up. In order to do so, the identical failure of the one component A has to be represented by two different basic events A and A^* . In complex fault trees this method is not feasible, is costly, and complicates clear analysis. Moreover, the probabilistic results are conservative as possible intersections between these events are not taken into account.

Qualitative and Probabilistic Calculation

At the end of the mission time each component’s failure probability and failure frequency equals those of the Boolean model from page 76.

One feature of the DFT approach is that the qualitative calculation of the failure function interprets the PAND gate as conventional AND gate. This certainly is a sensible conservative approach; as a consequence, though, the event sequence information is not present in the qualitative results. The failure function φ yields

$$\varphi_{\text{DFT 1}} = [A \wedge B] \vee [U \wedge A] \vee [E] \quad \text{and} \quad (6.11)$$

$$\varphi_{\text{DFT 2}} = [A^* \wedge B] \vee [U \wedge A] \vee [E] . \quad (6.12)$$

Isograph FaultTree+ provides the following results:

$$F_{\text{DFT 1}}(T_M) = 8,7933 \cdot 10^{-7} , \quad f_{\text{DFT 1}}(T_M) = 3,3946 \cdot 10^{-9} \frac{1}{h} \quad \text{and} \quad (6.13)$$

$$F_{\text{DFT 2}}(T_M) = 9,5962 \cdot 10^{-7} , \quad f_{\text{DFT 2}}(T_M) = 3,7967 \cdot 10^{-9} \frac{1}{h} . \quad (6.14)$$

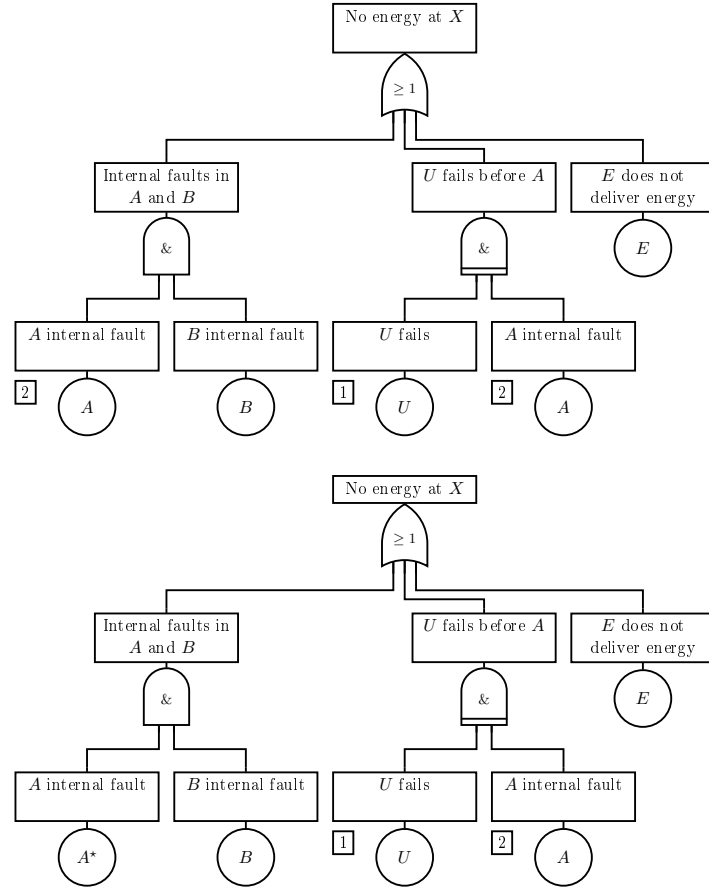


Figure 6.4: DFT fault trees in two versions, “DFT 1” (top) and “DFT 2” (bottom). In “DFT 1” event A is illegally meshed between the Boolean part of the fault tree and the dynamic module. In “DFT 2” the same real world failure of component A is represented by two different basic events A and A^* , which breaks the meshing. Both versions provide only approximative probabilistic results, though, and do not provide a qualitative analysis that also includes sequence information.

6.4 Comparison with Markov Diagrams

The example system’s markov model in this chapter is used as a reference for probabilistic calculations. Figure 6.5 shows the corresponding markov diagram, where all system failure states “no energy at X ” are denoted in bold. Event sequence information between U and A is taken into account.

Using $T_M = 400\text{h}$, Isograph FaultTree+ provides the following results:

$$F_{\text{MAR}}(T_M) = 9,5940 \cdot 10^{-7} , \quad f_{\text{MAR}}(T_M) = 3,7955 \cdot 10^{-9} \frac{1}{\text{h}} . \quad (6.15)$$

This modelling method does not allow for qualitative analysis like the analysis of minimal cutsets.

In comparison to the fault tree modelling methods from above the higher complexity of the markov method is apparent, which in real life inhibits the use of markov methods for analysis of many systems.

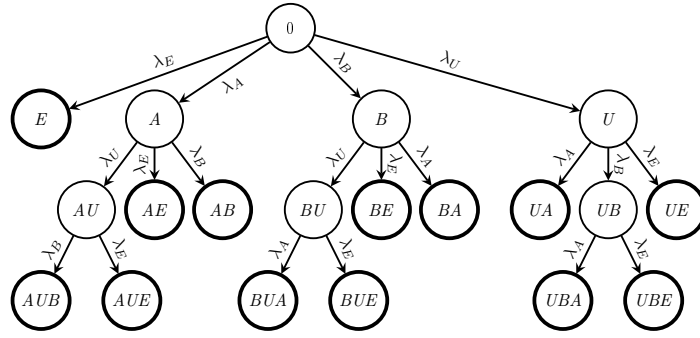


Figure 6.5: Markov Diagram (and also sequential failure tree) of the example system. System failure states “no energy at X ” are denoted in bold.

6.5 Dynamic FTA According to the TFTA Method

Figure 6.6 shows the temporal TFTA fault tree corresponding to the example system. One main benefit of the TFTA over the DFT approach is the way in which the fault tree structure is built. Just like the conventional Boolean FTA, it is possible to apply a “schematic-driven built-process”; i.e. to proceed backwards through the system, from its outputs to its inputs, and following the signal paths. This method is very intuitive as well as very systematic, thus reducing modelling errors. If there are meshings in the TFTA fault tree, they are broken up and resolved by the temporal logic. The same approach is generally not possible with the DFT because of its separated modules.

Qualitative and Probabilistic Calculation

The temporal system function of the temporal fault tree shown in figure 6.6 is given as

$$\begin{aligned}
 \varpi &= (A \vee E) \wedge (B \vee E \vee (U \vec{\wedge} A)) = \\
 &= [A \wedge B] \vee [A \wedge E] \vee [A \wedge (U \vec{\wedge} A)] \vee [E \wedge B] \vee [E] \vee [E \wedge (U \vec{\wedge} A)] = \\
 &= [A \wedge B] \vee [U \vec{\wedge} A] \vee [E] .
 \end{aligned} \tag{6.16}$$

Its three event sequences are already minimal according to chapter 4.3.2, as

$$[A \wedge B] \not\prec [U \vec{\wedge} A] \quad \text{arnd} \quad [A \wedge B] \not\prec [E] \quad \text{and} \quad [E] \not\prec [U \vec{\wedge} A] .$$

These event sequences are also MCSS and thus starting point for further qualitative evaluation. Qualitative analysis of the MCSS shows that the MCSS are indeed correctly calculated and do include the sequence information between events U and A . Further qualitative analysis then requires the transformation of the MCSS into a mutually exclusive (disjoint) form. The transformation according to chapter 4.3.3 yields an extended TDNF with mutually exclusive expressions:

$$\varpi = [\neg E \wedge (A \wedge B)] \vee [\neg B \neg E \wedge (U \vec{\wedge} A)] \vee [E] . \tag{6.17}$$

Using the components’ failure data from page 76, direct quantification is then possible:

$$\begin{aligned}
 F_{TFTA}(t) &= (1 - F_E(t)) \cdot F_A(t) \cdot F_B(t) + \\
 &\quad + (1 - F_E(t))(1 - F_B(t)) \cdot \int_0^t F_U(\tau) \cdot f_A(\tau) \cdot d\tau + F_E(t) ,
 \end{aligned}$$

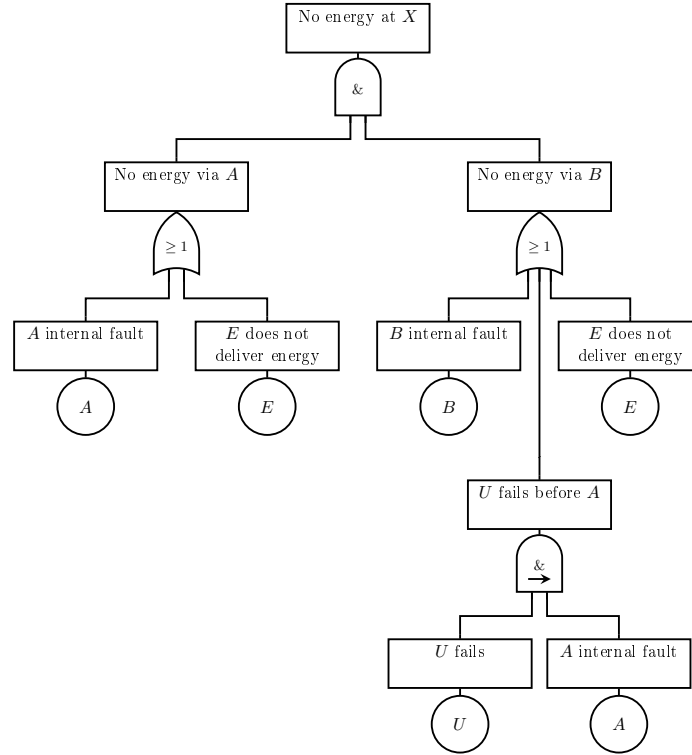


Figure 6.6: TFTA fault tree of the example system. It correctly takes the meshing of event A into account, as well as the sequence information between events U and A , and allows for a “schematic-driven built-process”. Probabilistically, the correct results are calculated, too.

$$F_{TFTA}(T_M) = 9,5940 \cdot 10^{-7} , \quad (6.18)$$

$$\begin{aligned} f_{TFTA}(t) &= (1 - F_E(t)) \cdot f_A(t) \cdot F_B(t) + (1 - F_E(t)) \cdot F_A(t) \cdot f_B(t) + \\ &\quad + (1 - F_E(t))(1 - F_B(t)) \cdot F_U(t) \cdot f_A(t) + f_E(t) \\ f_{TFTA}(T_M) &= 3,7955 \cdot 10^{-9} \frac{1}{h} . \end{aligned} \quad (6.19)$$

Comparison with the reference results from the markov model (see chapter 6.4) shows that the TFTA provides exact probabilistic results, too.

Approximation

Instead of using this exact calculation method, the TOP event’s failure parameters may also be approximated using the approach with reduced calculatory effort from chapter 5.5.

First, this approach is used on the extended TDNF of the temporal failure function from (6.17); $T_M = 400h$ then yields

$$\begin{aligned} F_{TFTA}(t) &\approx (1 - \lambda_E t) \cdot \lambda_A t \cdot \lambda_B t + \frac{1}{2}(1 - \lambda_E t)(1 - \lambda_B t) \cdot \lambda_U t \cdot \lambda_A t + \lambda_E t , \\ F_{TFTA}(T_M) &= 9,5984 \cdot 10^{-7} , \end{aligned} \quad (6.20)$$

$$\begin{aligned} f_{TFTA}(t) &\approx 2(1 - \lambda_E t) \cdot \lambda_A \cdot \lambda_B t + \frac{1}{2}(1 - \lambda_E t)(1 - \lambda_B t) \cdot \lambda_U \cdot \lambda_A t + \lambda_E , \\ f_{TFTA}(T_M) &= 3,7988 \cdot 10^{-7} \frac{1}{h} . \end{aligned} \quad (6.21)$$

Further significant simplification is possible using (6.16) instead of the temporal failure function from (6.17). The quantification of (6.16) yields

$$\begin{aligned} F_{TFTA}(t) &\approx \lambda_A t \cdot \lambda_B t + \frac{1}{2} \lambda_U t \cdot \lambda_A t + \lambda_E t, \\ F_{TFTA}(T_M) &\approx 9,6000 \cdot 10^{-7}, \end{aligned} \quad (6.22)$$

$$\begin{aligned} f_{TFTA}(t) &\approx 2 \lambda_A \cdot \lambda_B t + \frac{1}{2} \lambda_U \cdot \lambda_A t + \lambda_E, \\ f_{TFTA}(T_M) &\approx 3,8000 \cdot 10^{-7} \frac{1}{h}. \end{aligned} \quad (6.23)$$

On the one hand, it is no longer necessary to carry out the - possibly very costly - transformation into a disjoint form. On the other hand, the results are conservative approximations, usually good enough for at least a first assessment during a multi-step analysis.

Cutsets/Sequ.	Bool 1	Bool 2	DFT 1	DFT 2	Markov	TFTA
1.	E	E	E	E	–	E
2.	$A \wedge U$	U	$U \wedge A$	$U \wedge A$	–	$U \vec{\wedge} A$
3.	$A \wedge B$	$A \wedge B$	$A \wedge B$	$A^* \wedge B$	–	$A \wedge B$

Table 6.1: Comparison of the qualitative results of the different modelling methods for the example system from chapter 6.1. Minimal cutsets of the “Bool ...” and the “DFT ...” methods do not include event sequence information. As a consequence, there are failure combinations, that do not lead to a real life system failure, but are taken for system failures. The results of “Bool 2” and “DFT 2” deviate the most from the correct results represented by the MCSS of the TFTA. The markov model does not provide comparable qualitative results at all.

6.6 Summarizing the Results

The side-by-side comparison of Boolean FTA, DFT approach, markov model, and the new TFTA approach shows that the TFTA combines and surpasses the benefits of the other more conventional methods.

The TFTA adopts the basic steps of creating fault trees from the Boolean FTA. Most notably, it allows for a “schematic-driven built-process”; this assures a very systematic design and few modelling errors. The basic steps of the fault tree’s qualitative and probabilistic evaluation are also very similar between both methods. The failure function is qualitatively simplified into a minimal DNF; in a next step, this is then further qualitatively analysed, as well as transformed into mutually exclusive (disjoint) sub-expressions; these are then quantified. Other than the Boolean FTA, the TFTA takes relevant event sequence information into account qualitatively as well as probabilistically.

Looking at the qualitative results, only the TFTA provides minimal combinations of component failures that lead to a system failure, which include event sequence information, see table 6.1. The DFT and the conventional FTA provide minimal cutsets without event sequence information instead. Furthermore, the necessity of modules in the DFT is noteworthy: Meshing of events between Boolean and dynamic modules may lead to modelling errors which are difficult to discern and thus distort the qualitative results. It is possible to break such meshing up by

Method	$F(T_M) \quad [.] = 1$	$f(T_M) \quad [.] = \frac{1}{h}$	$\lambda(T_M) \quad [.] = \frac{1}{h}$
Bool 1	$1,3587 \cdot 10^{-6}$	$5,7899 \cdot 10^{-9}$	$5,7899 \cdot 10^{-9}$
Bool 2	$1,9986 \cdot 10^{-3}$	$4,9918 \cdot 10^{-6}$	$5,0019 \cdot 10^{-6}$
dynamic fault tree (DFT) 1	$8,7933 \cdot 10^{-7}$	$3,3946 \cdot 10^{-9}$	$3,3946 \cdot 10^{-9}$
DFT 2	$9,5962 \cdot 10^{-7}$	$3,7967 \cdot 10^{-9}$	$3,7967 \cdot 10^{-9}$
Markov	$9,5940 \cdot 10^{-7}$	$3,7955 \cdot 10^{-9}$	$3,7955 \cdot 10^{-9}$
temporal fault tree analysis (TFTA)	$9,5940 \cdot 10^{-7}$	$3,7955 \cdot 10^{-9}$	$3,7955 \cdot 10^{-9}$
TFTA (Approx. 1)	$9,5984 \cdot 10^{-7}$	$3,7988 \cdot 10^{-9}$	$3,7988 \cdot 10^{-9}$
TFTA (Approx. 2)	$9,6000 \cdot 10^{-7}$	$3,8000 \cdot 10^{-9}$	$3,8000 \cdot 10^{-9}$

Table 6.2: Comparison of the probabilistic results of the different modelling methods from chapters 6.2 to 6.5; the mission time is set to $T_M = 400h$. Obviously, the Boolean results are comparably conservative. The markov model is used as reference. The TFTA provides identical and therefore correct results, too. The last two rows show the results of the approximations of the probabilistic TFTA. “Approx 1” corresponds to the temporal failure function after it is transformed into a mutually exclusive (disjoint) form; “Approx 2” corresponds to the temporal failure function in a TDNF before being transformed into disjoint minterms, see (6.20) to (6.23).

using several “copied” events for one real world failure event; this provides good probabilistic approximations, but it reduces the significance and reliability of the qualitative results, as they contain nonsensical or even impossible event combinations.

The TFTA also provides correct probabilistic failure parameters at TOP event level; this is shown by comparison with the markov reference, see table 6.2. The Boolean models are comparatively conservative. The DFT provides correct results only for those fault trees that do not have events meshed between Boolean and dynamic modules. If such meshings are necessary, then the DFT usually provides optimistic (i.e. too small) probabilistic results.

The TFTA is also well suited for a multi-step approach of modelling, where the results’ accuracy is improved step by step. The TFTA’s approach with reduced calculatory effort provides conservative probabilistic approximations as well as, qualitatively, the minimal failure sequences.

7 TFTA Analysis of an Automotive ECU Architecture

Insight separated from practice
remains ineffective.

(Erich Fromm)

This chapter uses the TFTA method on a more complex example and shows how TFTA may be applied to more than academic minimal examples.

7.1 The Example System

The example system in figure 7.1 is an abstraction of a system architecture typically used in the automotive domain for safety critical systems up to SIL 3 according to IEC 61508 or ASIL D according to ISO 26262.

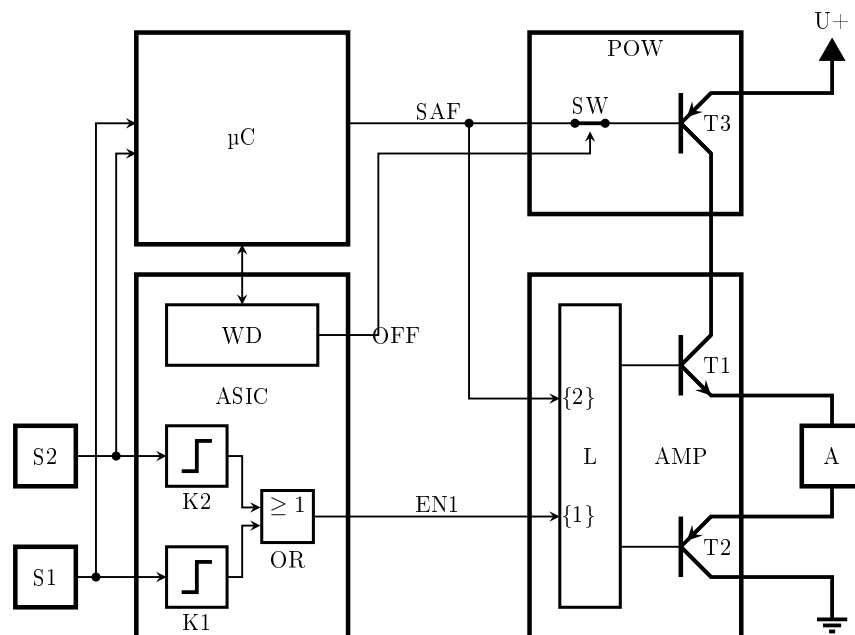


Figure 7.1: A real world example systems which is analyzed using the TFTA method.

The structure of this chapter: In chapter 7.2 the temporal fault tree corresponding to the example system is shown. The qualitative analysis in chapter 7.3 and the probabilistic evaluation in chapter 7.4 are followed by a discussion of the results in chapter 7.5.

7.1.1 System Description, Safety Goal and Safe State

Scope

The example system consists of the components and signals wlisted in table 7.1.

Component	Subcomponent	Description
S1		sensor 1
S2		sensor 2
μC		microcontroller
ASIC		system-ASIC
	WD	watchdog for μC
	K1	comparator 1
	K2	comparator 2
	OR	OR gate
POW		power switch
	SW	emergency switch
	T3	power transistor
AMP		driver IC
	L	logic
	T1	high side power stage
	T2	low side power stage
A		actuator
Signal	Description	
EN	enable signal for the logic in the driver IC	
SAF	enable signal for power transistor and driver IC	
OFF	disable/cutoff signal from watchdog	

Table 7.1: Components and signals of the example system in figure 7.1

Functional Description and Safety Concept

The example system is used to safely activate actuator A based on some sensor information. The actuator shall be activated, if (and only if) the sensor input shows that some threshold level is exceeded. If the sensor input is below this threshold, the actuator shall be deactivated. The system includes several redundancy measures in order to increase its functional safety.

Both sensors S1 and S2 record some physical parameters from the surrounding. Each sensor sends its data over a separate serial port to microcontroller μC as well as the system ASIC. The transmission is protected using CRC and alive counters.

Microcontroller μC evaluates the sensor data of both sensors S1 and S2. If at least one of the sensors' data is below the threshold, output SAF of the μC is deactivated. If both of the sensors' data are above the threshold, μC activates the power transistor T3 via the SAF signal. At the same time, μC activates the power stages T1 and T2 in the AMP driver via AMP's enabler input {2}. Meanwhile, the microcontroller serves the intelligent watchdog in the system ASIC via an additional bidirectional port.

The system ASIC evaluates the same sensor data as the microcontroller. It has two hardware comparators K1 and K2. Comparator K1 evaluates data from sensor S1. Comparator K2

evaluates data from sensor S2. If at least one of the hardware comparators detects that the corresponding threshold is exceeded, it activates its output EN. Additionally, the system ASIC includes an intelligent watchdog WD. Using several mechanisms, the watchdog monitors that the μ C hardware is operable and the operating system and the application software on μ C run correctly. This is accomplished, first, using a window watchdog triggered by special waypoints within the program software; second, WD queries μ C and monitors the provided answers. If μ C answers too early or too late or provides a wrong answer, WD activates (opens) a separate emergency switch SW via the OFF signal. If SW is open, T3 is deactivated independently of SAF; the power supply to the power stages and thus to the actuator is interrupted.

Driver AMP consists of the two power stages T1 and T2 as well as an internal logic L. L activates the power stages, if (and only if) enable input {1} is activated first, and then enable input {2} is activated second. Every other sequence does not activate the power stages.

Normally, the activation abides the sequence {1}, {2}: on the one hand, data from S1 and S2 do not occur at exactly the same time, e.g. because S1 and S2 are spatially separated. Then, signal EN will always be activated first, when the first sensor data indicates an exceeding of the threshold. On the other hand, the software in μ C also carries some latency to EN, which leads to an internally delayed activation of SAF.

Safety Goal, Safe State, and Fault Tolerance Time Span

The system's hazard and risk analysis yields the following safety goal: "prevent erroneous current feed through the actuator". The corresponding safety state is "no current feed through actuator". The fault tolerant time span is 0 seconds, i.e. current feeds are considered immediately dangerous and are thus not allowed even for very short times.

7.1.2 Failures

Using the simplification that all connections between components S1, S2, μ C, K1, K2, WD, SW, T1, T2, T3, L, and A are ideal and have no faults, the components' failures listed in table 7.2 remain. The failures' dangerousness depends on their potential to contribute to an infraction of the safety goal. The listed safety measures prevent a direct infraction of the safety goal by the failures. For a dynamic failure analysis two areas of the system are specifically interesting. First, there is a sequence logic in L, and second there are dangerous failures of WD and SW (numbers 18 and 27 in table 7.2, respectively) in combination with a failure of the microcontroller. These failures of the watchdog or switch SW are relevant, if (and only if) at least one of them occurs before failures of μ C. But if μ C fails first, while WD as well as SW are operational, i.e. have not failed, or have failed, but "in a safe direction", it is assumed, that this was detected and thus the system is disabled. Further dangerous consequences are then ruled out. Furthermore, dependent failures, and especially common cause failures (CCF), are not considered in this example.

Failures of μ C may not be easily attributed to specific hardware faults, as μ C's functionality is largely realised in software. It is assumed, that the different failures of μ C – numbers 9 to 17 in table 7.2 – occur independent from each other.

7.2 Temporal Fault Tree

A temporal fault tree for the example system is to be created. It shall provide evidence that no dangerous single failure leads to a direct infraction of the safety goal; this is called "single failure

Comp.	Nr.	Failure	Failure consequence	Dangerous	Prevention against direct infraction of the safety goal
S1	1	wrongly provide value above threshold	µC and ASIC recognize activation criterion	yes	A is activated only if second fault in S2
	2	wrongly provide value below threshold	µC and ASIC don't recognize activation criterion	no	
	3	no communication with µC	µC doesn't recognize activation criterion	no	
	4	no communication with ASIC	ASIC may enable EN with only S2	no	
S2	5	wrongly provide value above threshold	µC and ASIC recognize activation criterion	yes	A is activated only if second fault in S1
	6	wrongly provide value below threshold	µC and ASIC don't recognize activation criterion	no	
	7	no communication with µC	µC doesn't recognize activation criterion	no	
	8	no communication with ASIC	ASIC may enable EN with only S1	no	
µC	9	µC stuck-at failure	µC can't change output SAF and can't serve WD	no	WD detects µC failure and activates cutoff
	10	address, program counter, or IO-failure	µC arbitrarily changes SAF output;	yes	
	11	input S1 stuck-at	WD is not correctly served	no	
	12	input S2 stuck-at	no evaluation of S1	no	
	13	data from S1 is wrongly interpreted	µC recognizes activation criterion	yes	A is activated only if second fault in S2
	14	to be above threshold	µC doesn't recognize activation criterion	no	
	15	data from S1 is wrongly interpreted	µC recognizes activation criterion	yes	
	16	to be below threshold	µC doesn't recognize activation criterion	no	
	17	data from S2 is wrongly interpreted	µC doesn't recognize activation criterion	no	A is activated only if second fault in S1
	18	to be above threshold	µC recognizes activation criterion	yes	
	19	data from S2 is wrongly interpreted	µC doesn't recognize activation criterion	no	
	20	to be below threshold	µC recognizes activation criterion	yes	
WD	21	data from S2 is wrongly interpreted	WD doesn't recognize activation criterion	no	A is activated only if second fault in AMP, or EN occurs before SAF activated
	22	to be above threshold	WD does not activated Signal OFF and POW-SW	yes	
	23	data from S2 is wrongly interpreted	WD activates signal OFF and cutoff signal,	no	
	24	to be below threshold	no supply of AMP (= safe state)	yes	
K1	25	wrongly interpret data from S1 as above threshold	ASIC activates EN	yes	A is activated only if second fault in µC or AMP
	26	wrongly interpret data from S1 as below threshold	ASIC does not activate EN	no	
	27	wrongly interpret data from S2 as above threshold	ASIC activates EN	yes	
	28	wrongly interpret data from S2 as below threshold	ASIC does not activate EN	no	
K2	29	wrongly decide to activate without request by ASIC-K1 or K2	ASIC does not activate EN	no	A is activated only if second fault in µC or AMP
	30	wrongly ignore activation request by ASIC-K1 or K2	no activation of T3 when SAF is activated	yes	
	31	open cutoff wrongly without request	high side power stage T1 is supplied with energy	yes	
	32	not open cutoff despite OFF signal	no energy supply to T1	no	
SW	33	switch on without request by SAF	T3 is connected to actuator A	yes	A is activated only if T2 is also wrongly switched on and with another multiple point fault in ASIC or AMP
	34	switch on without request by AMP-L	A is not supplied with energy	no	
	35	not switch on despite request by AMP-L	actuator A is connected to ground	yes	
	36	switch on without request by AMP-L	A has no connection to ground	no	
T1	37	not switch on despite request by AMP-L	see 30	yes	A is activated only if T1 is also wrongly switched on and another multiple point fault in µC or POW
	38	switch on without request by AMP-L	see 31	no	
	39	not switch on despite request by AMP-L	see 32	yes	
	40	switch on without request by AMP-L	see 33	no	
T2	41	not switch on despite request by AMP-L	T3 is connected to actuator A and actuator A is connected to ground	yes	A is activated only in combination with a second fault in µC or POW
	42	switch on without request by AMP-L	safe state	no	
	43	not switch on despite request by AMP-L			
	44	switch on without request by AMP-L			
L	45	not switch on despite request by AMP-L			
	46	activate AMP-T1 without request			
	47	not activate AMP-T1 despite request			
	48	activate AMP-T2 without request			
A	49	not activate AMP-T2 despite request			
	50	activate AMP-T1 and T2 without request			
	51	no action despite correct energy supply			
	52				

Table 7.2: Overview over the possible failures of the example system from figure 7.1.

resistance”. Furthermore, an MCSS analysis shall provide the most relevant combinations of dangerous failures. A probabilistic quantification shall then provide evidence that the system’s failure rate stays below the threshold as defined for ASIL D in ISO 26262.

The TOP event of the fault tree is the “infraction of the safety goal”, i.e. the “erroneous current feed through the actuator”. As the system has time-dependencies between its components’ failures, it is necessary to use temporal fault tree gates. Figures 7.2 bis 7.4 show the temporal fault tree for the example system, split into three parts. The basic events’ numbers correspond to those in table 7.2.

In total the temporal fault tree consists of 32 gates and 34 basic events. There are 16 meshed gates and 18 meshed basic events. Two of the gates are PAND gates, which appear three times because of meshings. These temporal gates represent sub fault trees with ten different basic events and ten different gates.

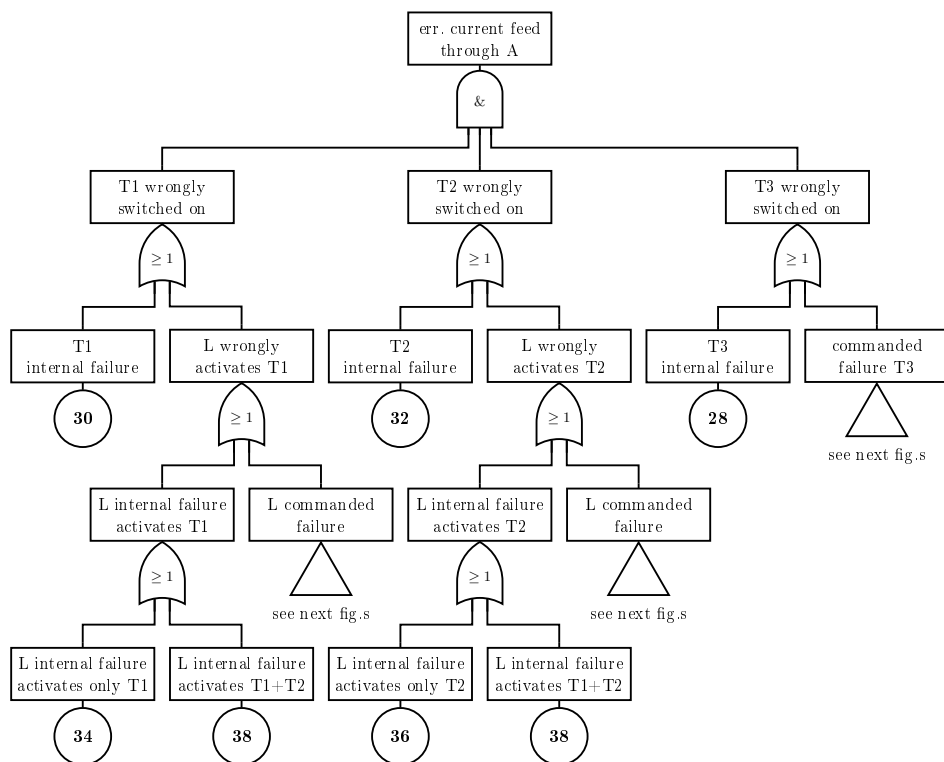


Figure 7.2: Fault tree of the example system from figure 7.1, part 1. The basic events’ numbers correspond to those in table 7.2.

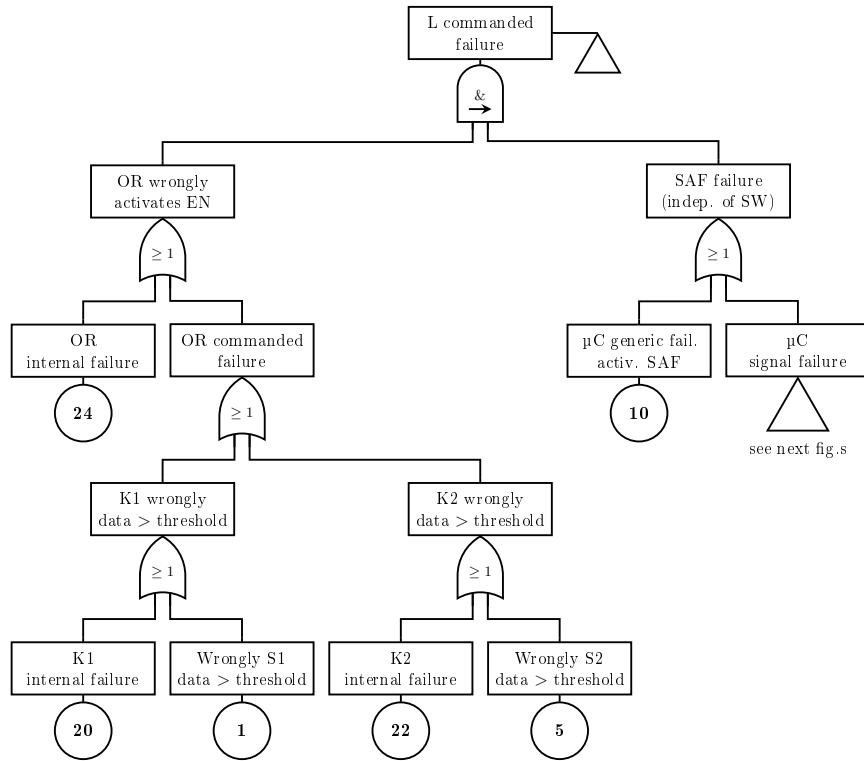


Figure 7.3: Fault tree of the example system from figure 7.1, part 2.

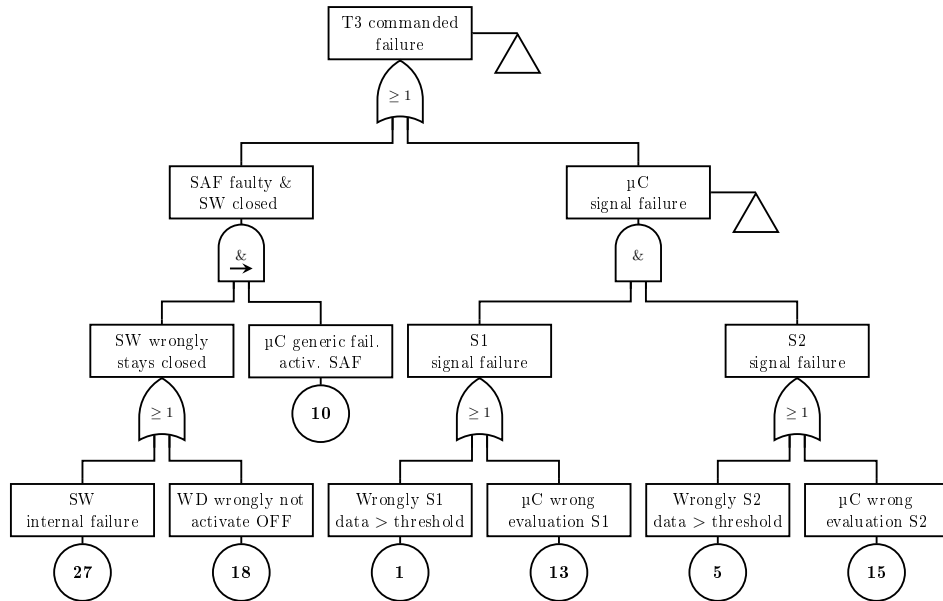


Figure 7.4: Fault tree of the example system from figure 7.1, part 3.

7.3 Qualitative Analysis of the Temporal Fault Tree

7.3.1 Temporal Failure Function

The failure function for the TOP event is directly read from the fault tree in figures 7.2 to 7.4:

$$\begin{aligned}
 \varpi = & \left(X_{30} \vee X_{34} \vee X_{38} \vee \left[\underbrace{(X_{24} \vee X_{20} \vee X_1 \vee X_{22} \vee X_5)}_A \vec{\wedge} \right. \right. \\
 & \left. \left. \vec{\wedge} \left(\underbrace{X_{10} \vee [(X_1 \vee X_{13}) \wedge (X_5 \vee X_{15})]}_B \right) \right] \right) \wedge \\
 & \wedge \left(X_{32} \vee X_{36} \vee X_{38} \vee \left[A \vec{\wedge} B \right] \right) \wedge \\
 & \wedge \left(\underbrace{X_{28} \vee [(X_1 \vee X_{13}) \wedge (X_5 \vee X_{15})] \vee [(X_{27} \vee X_{18}) \vec{\wedge} X_{10}]}_C \right) . \tag{7.1}
 \end{aligned}$$

Substitutions A , B , and C facilitate further simplification:

$$\begin{aligned}
 \varpi = & \left(X_{30} \vee X_{34} \vee X_{38} \vee [A \vec{\wedge} B] \right) \wedge \left(X_{32} \vee X_{36} \vee X_{38} \vee [A \vec{\wedge} B] \right) \wedge (C) = \\
 = & [X_{30} \wedge X_{32} \wedge C] \vee [X_{30} \wedge X_{36} \wedge C] \vee [X_{30} \wedge X_{38} \wedge C] \vee [X_{30} \wedge (A \vec{\wedge} B) \wedge C] \vee \\
 & \vee [X_{34} \wedge X_{32} \wedge C] \vee [X_{34} \wedge X_{36} \wedge C] \vee [X_{34} \wedge X_{38} \wedge C] \vee [X_{34} \wedge (A \vec{\wedge} B) \wedge C] \vee \\
 & \vee [X_{38} \wedge X_{32} \wedge C] \vee [X_{38} \wedge X_{36} \wedge C] \vee [X_{38} \wedge X_{38} \wedge C] \vee [X_{38} \wedge (A \vec{\wedge} B) \wedge C] \vee \\
 & \vee [(A \vec{\wedge} B) \wedge X_{32} \wedge C] \vee [(A \vec{\wedge} B) \wedge X_{36} \wedge C] \vee [(A \vec{\wedge} B) \wedge X_{38} \wedge C] \vee \\
 & \vee [(A \vec{\wedge} B) \wedge (A \vec{\wedge} B) \wedge C] . \tag{7.2}
 \end{aligned}$$

Applying the laws of absorption and idempotency yields

$$\begin{aligned}
 \varpi = & [X_{30} \wedge X_{32} \wedge C] \vee [X_{30} \wedge X_{36} \wedge C] \vee [X_{34} \wedge X_{32} \wedge C] \vee \\
 & \vee [X_{34} \wedge X_{36} \wedge C] \vee [X_{38} \wedge C] \vee [(A \vec{\wedge} B) \wedge C] . \tag{7.3}
 \end{aligned}$$

The next chapter transforms the temporal failure function from (7.3) according to the laws of temporal logic. The analysis of the resulting MCSS of ϖ follows in chapter 7.3.3.

7.3.2 Transformation According to the Temporal Logic Rules

MCSS of the First Five Terms in (7.3):

The temporal failure function in (7.3) has five parts

$$[X_{30} \wedge X_{32} \wedge C], [X_{30} \wedge X_{36} \wedge C], [X_{34} \wedge X_{32} \wedge C], [X_{34} \wedge X_{36} \wedge C], [X_{38} \wedge C] \tag{7.4}$$

that have no reference to event A . Basic events $X_{30}, X_{32}, X_{34}, X_{36}, X_{38}$ are not also included in C . If each of these five expressions is combined with the TDNF of C , i.e.

$$C = X_{28} \vee (X_1 X_5) \vee (X_1 X_{15}) \vee (X_5 X_{13}) \vee (X_{13} X_{15}) \vee (X_{27} \vec{\wedge} X_{10}) \vee (X_{18} \vec{\wedge} X_{10}) , \tag{7.5}$$

They provide nine different event sequences each, as shown here for the one example with $X_{38} \wedge C$:

$$X_{38} \wedge C = X_{38} \wedge [X_{28} \vee (X_1 X_5) \vee (X_1 X_{15}) \vee (X_5 X_{13}) \vee (X_{13} X_{15}) \vee$$

$$\vee (X_{27} \vec{\wedge} X_{10}) \vee (X_{18} \vec{\wedge} X_{10})] . \quad (7.6)$$

Next, this provides five event sequences each, like in

$$[X_{38}X_1X_5], [X_{38}X_1X_{15}], [X_{38}X_5X_{13}], [X_{38}X_{13}X_{15}], [X_{28}X_{38}] . \quad (7.7)$$

Furthermore, there are four additional event sequences (without SAND) from $X_{38} \wedge (X_{18} \vec{\wedge} X_{10})$ und $X_{38} \wedge (X_{27} \vec{\wedge} X_{10})$:

$$[(X_{18}X_{38}) \vec{\wedge} X_{10}], [X_{18} \vec{\wedge} X_{10} \vec{\wedge} X_{38}], [(X_{27}X_{38}) \vec{\wedge} X_{10}], [X_{27} \vec{\wedge} X_{10} \vec{\wedge} X_{38}] . \quad (7.8)$$

In total there are 45 event sequences, as shown in table 7.3.

(extended) MCSS of rank two:	
1: $X_{28}X_{38}$	
(extended) MCSS of rank three:	
1: $X_{28}X_{30}X_{32}$	7: $X_{28}X_{30}X_{36}$
2: $X_{28}X_{32}X_{34}$	8: $X_{28}X_{34}X_{36}$
3: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} X_{38}$	9: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} X_{38}$
4: $(X_{18}X_{38}) \vec{\wedge} X_{10}$	10: $(X_{27}X_{38}) \vec{\wedge} X_{10}$
5: $X_{38}X_1X_5$	11: $X_{38}X_1X_{15}$
6: $X_{38}X_5X_{13}$	12: $X_{38}X_{13}X_{15}$
(extended) MCSS of rank four:	
1: $X_1X_5X_{30}X_{32}$	17: $X_1X_5X_{30}X_{36}$
2: $X_1X_{15}X_{30}X_{32}$	18: $X_1X_{15}X_{30}X_{36}$
3: $X_{13}X_5X_{30}X_{32}$	19: $X_{13}X_5X_{30}X_{36}$
4: $X_{13}X_{15}X_{30}X_{32}$	20: $X_{13}X_{15}X_{30}X_{36}$
5: $X_1X_5X_{32}X_{34}$	21: $X_1X_5X_{34}X_{36}$
6: $X_1X_{15}X_{32}X_{34}$	22: $X_1X_{15}X_{34}X_{36}$
7: $X_{13}X_5X_{32}X_{34}$	23: $X_{13}X_5X_{34}X_{36}$
8: $X_{13}X_{15}X_{32}X_{34}$	24: $X_{13}X_{15}X_{34}X_{36}$
9: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} (X_{30}X_{32})$	25: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} (X_{30}X_{32})$
10: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} (X_{30}X_{36})$	26: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} (X_{30}X_{36})$
11: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} (X_{32}X_{34})$	27: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} (X_{32}X_{34})$
12: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} (X_{34}X_{36})$	28: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} (X_{34}X_{36})$
13: $(X_{18}X_{30}X_{32}) \vec{\wedge} X_{10}$	29: $(X_{27}X_{30}X_{32}) \vec{\wedge} X_{10}$
14: $(X_{18}X_{30}X_{32}) \vec{\wedge} X_{10}$	30: $(X_{27}X_{30}X_{32}) \vec{\wedge} X_{10}$
15: $(X_{18}X_{32}X_{34}) \vec{\wedge} X_{10}$	31: $(X_{27}X_{32}X_{34}) \vec{\wedge} X_{10}$
16: $(X_{18}X_{34}X_{36}) \vec{\wedge} X_{10}$	32: $(X_{27}X_{34}X_{36}) \vec{\wedge} X_{10}$

Table 7.3: MCSS of ranks two, three, and four, resulting from the first five expressions in (7.3).

Simplification of $A \vec{\wedge} B$:

First, $A \vec{\wedge} B$ has to be broken apart. Because of limited space in this thesis, only the first transformational steps are shown, as relevant for understanding the basic concept. B may be transformed into the following DNF:

$$B = X_{10} \vee (X_1X_5) \vee (X_1X_{15}) \vee (X_5X_{13}) \vee (X_{13}X_{15}) = X_{10} \vee \eta . \quad (7.9)$$

According to the temporal distributive law for temporal expressions of type I – see (4.76) –,

$$A \vec{\wedge} B = [\neg \eta \wedge (A \vec{\wedge} X_{10})] \vee [\neg X_{10} \wedge (A \vec{\wedge} \eta)] \vee [A \vec{\wedge} (X_{10} \vec{\wedge} \eta)] =$$

$$\begin{aligned}
&= [\neg(X_1X_5 \vee X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15}) \wedge (A \vec{\wedge} X_{10})] \vee \\
&\quad \vee [\neg X_{10} \wedge (A \vec{\wedge} (X_1X_5 \vee X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15}))] \vee \\
&\quad \vee [A \vec{\wedge} (X_{10} \bar{\wedge} (X_1X_5 \vee X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15}))] = \\
&= \eta_1 \vee \eta_2 \vee \eta_3 .
\end{aligned} \tag{7.10}$$

Expression η_1 may then easily be transformed into a TDNF:

$$\begin{aligned}
\eta_1 &= \neg(X_1X_5 \vee X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15}) \wedge (A \vec{\wedge} X_{10}) = \\
&= [(\neg X_1 \neg X_{13}) \wedge (A \vec{\wedge} X_{10})] \vee [(\neg X_5 \neg X_{15}) \wedge (A \vec{\wedge} X_{10})] .
\end{aligned} \tag{7.11}$$

Expression η_2 is more complex and thus is transformed step by step:

$$\begin{aligned}
\eta_2 &= \neg X_{10} \wedge \left(\neg(X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15}) \wedge (A \vec{\wedge} (X_1X_5)) \right) \vee \\
&\quad \vee \neg X_{10} \wedge \left(\neg(X_1X_5) \wedge (A \vec{\wedge} (X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15})) \right) \vee \\
&\quad \vee \neg X_{10} \wedge \left(A \vec{\wedge} ((X_1X_5) \bar{\wedge} (X_1X_{15} \vee X_5X_{13} \vee X_{13}X_{15})) \right) = \\
&= \eta_{2a} \vee \eta_{2b} \vee \eta_{2c} .
\end{aligned} \tag{7.12}$$

The first expression in (7.12) provides three event sequences –

$$\begin{aligned}
\eta_{2a} &= \neg X_{10} \wedge [(\neg X_1 \neg X_{13}) \vee (\neg X_5 \neg X_{15}) \vee (\neg X_{13} \neg X_{15})] \wedge (A \vec{\wedge} (X_1X_5)) = \\
&= [(\neg X_1 \neg X_{10} \neg X_{13}) \wedge (A \vec{\wedge} (X_1X_5))] \vee \\
&\quad \vee [(\neg X_5 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_1X_5))] \vee \\
&\quad \vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (A \vec{\wedge} (X_1X_5))] =
\end{aligned} \tag{7.13}$$

but only the third of these does not yield *False*, if rules (4.51) and (4.52) are applied.

Therefore,

$$\eta_{2a} = [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (A \vec{\wedge} (X_1X_5))] . \tag{7.14}$$

The second expression in (7.12) itself provides three expressions:

$$\begin{aligned}
\eta_{2b} &= (\neg X_{10} \neg (X_1X_5)) \wedge \left(\neg(X_5X_{13} \vee X_{13}X_{15}) \wedge (A \vec{\wedge} (X_1X_{15})) \right) \vee \\
&\quad \vee (\neg X_{10} \neg (X_1X_5)) \wedge \left(\neg(X_1X_{15}) \wedge (A \vec{\wedge} (X_5X_{13} \vee X_{13}X_{15})) \right) \vee \\
&\quad \vee (\neg X_{10} \neg (X_1X_5)) \wedge \left(A \vec{\wedge} ((X_1X_{15}) \bar{\wedge} (X_5X_{13} \vee X_{13}X_{15})) \right) = \\
&= \eta_{2b1} \vee \eta_{2b2} \vee \eta_{2b3} .
\end{aligned} \tag{7.15}$$

Using the rules in (4.51) and (4.52) on

$$\begin{aligned}
\eta_{2b1} &= (\neg X_{10} \neg (X_1X_5)) \wedge [(\neg X_{13} \vee [\neg X_5 \neg X_{15}]) \wedge (A \vec{\wedge} (X_1X_{15}))] = \\
&= [(\neg X_1 \neg X_{10} \neg X_{13}) \wedge (A \vec{\wedge} (X_1X_{15}))] \vee \\
&\quad \vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (A \vec{\wedge} (X_1X_{15}))] \vee \\
&\quad \vee [(\neg X_1 \neg X_5 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_1X_{15}))] \vee \\
&\quad \vee [(\neg X_5 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_1X_{15}))]
\end{aligned} \tag{7.16}$$

leaves only

$$\eta_{2b1} = [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (A \vec{\wedge} (X_1 X_{15}))] . \quad (7.17)$$

The second part of (7.15) again provides three expressions, i.e.

$$\begin{aligned} \eta_{2b2} = & (\neg X_{10} \wedge (\neg X_1 \vee [\neg X_5 \neg X_{15}])) \wedge (\neg (X_{13} X_{15}) \wedge (A \vec{\wedge} (X_5 X_{13}))) \vee \\ & \vee (\neg X_{10} \wedge (\neg X_1 \vee [\neg X_5 \neg X_{15}])) \wedge (\neg (X_5 X_{13}) \wedge (A \vec{\wedge} (X_{13} X_{15}))) \vee \\ & \vee (\neg X_{10} \wedge (\neg X_1 \vee [\neg X_5 \neg X_{15}])) \wedge (A \vec{\wedge} ((X_5 X_{13}) \bar{\wedge} (X_{13} X_{15}))) = \\ = & \eta_{2b2a} \vee \eta_{2b2b} \vee \eta_{2b2c} . \end{aligned} \quad (7.18)$$

Because of rules (4.51) and (4.52), the first of these expressions may be simplified to

$$\begin{aligned} \eta_{2b2a} = & [\neg X_{10} \wedge (\neg X_1 \vee [\neg X_5 \neg X_{15}]) \wedge \neg (X_{13} X_{15})] \wedge (A \vec{\wedge} (X_5 X_{13})) = \\ = & (\neg X_1 \neg X_{10} \neg X_{13}) \wedge (A \vec{\wedge} (X_5 X_{13})) \vee \\ & \vee (\neg X_1 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_5 X_{13})) \vee \\ & \vee (\neg X_5 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_5 X_{13})) \vee = \\ = & [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_5 X_{13}))] . \end{aligned} \quad (7.19)$$

The same steps repeated for the second expression yield

$$\begin{aligned} \eta_{2b2b} = & [\neg X_{10} \wedge (\neg X_1 \vee [\neg X_5 \neg X_{15}]) \wedge \neg (X_5 X_{13})] \wedge (A \vec{\wedge} (X_{13} X_{15})) = \\ = & [(\neg X_1 \neg X_5 \neg X_{10}) \wedge (A \vec{\wedge} (X_{13} X_{15}))] . \end{aligned} \quad (7.20)$$

Because of

$$\begin{aligned} (X_5 X_{13}) \bar{\wedge} (X_{13} X_{15}) = & [(X_5 X_{15}) \vec{\wedge} X_{13}] \vee [X_{13} \vec{\wedge} (X_5 \bar{\wedge} X_{15})] \vee [X_{15} \vec{\wedge} (X_5 \bar{\wedge} X_{13})] \vee \\ & \vee [X_5 \vec{\wedge} (X_{13} \bar{\wedge} X_{15})] \vee [X_5 \bar{\wedge} X_{13} \bar{\wedge} X_{15}] \end{aligned} \quad (7.21)$$

the third expression in (7.18) provides

$$\begin{aligned} \eta_{2b2c} = & (\neg X_1 \neg X_{10}) \wedge (A \vec{\wedge} [(X_5 X_{15}) \vec{\wedge} X_{13}] \vee A \vec{\wedge} [X_{13} \vec{\wedge} (X_5 \bar{\wedge} X_{15})] \vee \\ & \vee A \vec{\wedge} [X_{15} \vec{\wedge} (X_5 \bar{\wedge} X_{13})] \vee A \vec{\wedge} [X_5 \vec{\wedge} (X_{13} \bar{\wedge} X_{15})] \vee A \vec{\wedge} [X_5 \bar{\wedge} X_{13} \bar{\wedge} X_{15}]) , \end{aligned} \quad (7.22)$$

but only event sequence $(\neg X_1 \neg X_{10}) \wedge [(A \wedge X_5 \wedge X_{15}) \vec{\wedge} X_{13}]$ is free of SANDs. Therefore, only this one event sequence is taken into account, as in this example dependent failures are not considered, see chapter 7.1.2.

Inserting (7.22) and (7.20) and (7.19) into (7.18) provides three event sequences

$$\begin{aligned} \eta_{2b2} = & [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_5 X_{13}))] \vee \\ & \vee [(\neg X_1 \neg X_5 \neg X_{10}) \wedge (A \vec{\wedge} (X_{13} X_{15}))] \vee \\ & \vee [(\neg X_1 \neg X_{10}) \wedge ((A \wedge X_5 \wedge X_{15}) \vec{\wedge} X_{13})] . \end{aligned} \quad (7.23)$$

The third expression from (7.15) is still open. Using the same steps, it may be simplified to

$$\eta_{2b3} = (\neg X_{10} \neg (X_1 X_5)) \wedge (A \vec{\wedge} ((X_1 X_{15}) \bar{\wedge} (X_5 X_{13} \vee X_{13} X_{15}))) =$$

$$\begin{aligned}
&= (\neg X_{10} \neg(X_1 X_5)) \wedge \left(\neg(X_{13} X_{15}) \wedge (A \vec{\wedge} ((X_1 X_{15}) \bar{\wedge} (X_5 X_{13}))) \right) \vee \\
&\quad \vee (\neg X_{10} \neg(X_1 X_5)) \wedge \left(\neg(X_5 X_{13}) \wedge (A \vec{\wedge} ((X_1 X_{15}) \bar{\wedge} (X_{13} X_{15}))) \right) \vee \\
&\quad \vee (\neg X_{10} \neg(X_1 X_5)) \wedge \left(A \vec{\wedge} ((X_1 X_{15}) \bar{\wedge} (X_5 X_{13}) \bar{\wedge} (X_{13} X_{15})) \right).
\end{aligned}$$

Applying rules (4.51) and (4.52) provides a simplified η_{2b3} :

$$\begin{aligned}
\eta_{2b3} &= False \vee [(\neg X_{10} \neg(X_1 X_5) \neg(X_5 X_{13})) \wedge ((A \wedge X_1 \wedge X_{13}) \vec{\wedge} X_{15})] \vee False = \\
&= [(\neg X_5 \neg X_{10}) \wedge ((A \wedge X_1 \wedge X_{13}) \vec{\wedge} X_{15})] .
\end{aligned} \tag{7.24}$$

The results in (7.24) and (7.23) and (7.17) are inserted into (7.15), which provides the five event sequences (again without SANDs) of η_{2b} .

Transformation of expressions η_{2c} and η_2 and η_3 is carried out analogously to the detailed steps from above. This is not described explicitly.

Expression η_{2c} from (7.12) provides two expressions (again without SAND):

$$\begin{aligned}
\eta_{2c} &= [(\neg X_{10} \neg X_{15}) \wedge ((A \wedge X_1 \wedge X_{13}) \vec{\wedge} X_5)] \vee \\
&\quad \vee [(\neg X_{10} \neg X_{13}) \wedge ((A \wedge X_5 \wedge X_{15}) \vec{\wedge} X_1)] .
\end{aligned} \tag{7.25}$$

Together with (7.14) and (7.15) η_2 therefore yields eight event sequences (without SAND).

Then, expression η_3 provides only event sequences with at least one SAND and is therefore not considered further.

In total, $A \vec{\wedge} B$ therefore yields two event sequences without SAND from η_1 , see (7.11), and eight event sequences from η_2 :

$$\begin{aligned}
A \vec{\wedge} B &= [(\neg X_1 \neg X_{13}) \wedge (A \vec{\wedge} X_{10})] \vee && \langle \text{ES1} \rangle \\
&\vee [(\neg X_5 \neg X_{15}) \wedge (A \vec{\wedge} X_{10})] \vee && \langle \text{ES2} \rangle \\
&\vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (A \vec{\wedge} (X_1 X_5))] \vee && \langle \text{ES3} \rangle \\
&\vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (A \vec{\wedge} (X_1 X_{15}))] \vee && \langle \text{ES4} \rangle \\
&\vee [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (A \vec{\wedge} (X_5 X_{13}))] \vee && \langle \text{ES5} \rangle \\
&\vee [(\neg X_1 \neg X_5 \neg X_{10}) \wedge (A \vec{\wedge} (X_{13} X_{15}))] \vee && \langle \text{ES6} \rangle \\
&\vee [(\neg X_1 \neg X_{10}) \wedge ((A \wedge X_5 \wedge X_{15}) \vec{\wedge} X_{13})] \vee && \langle \text{ES7} \rangle \\
&\vee [(\neg X_5 \neg X_{10}) \wedge ((A \wedge X_1 \wedge X_{13}) \vec{\wedge} X_{15})] \vee && \langle \text{ES8} \rangle \\
&\vee [(\neg X_{10} \neg X_{15}) \wedge ((A \wedge X_1 \wedge X_{13}) \vec{\wedge} X_5)] \vee && \langle \text{ES9} \rangle \\
&\vee [(\neg X_{10} \neg X_{13}) \wedge ((A \wedge X_5 \wedge X_{15}) \vec{\wedge} X_1)] . && \langle \text{ES10} \rangle
\end{aligned} \tag{7.26}$$

Below, identifiers $\langle \text{ES1} \rangle$ to $\langle \text{ES10} \rangle$ are used as a reference to the respective event sequence. The transformation of A is done using the temporal distributive law for temporal expressions of type II according to (4.78). Applying (7.26) and further simplification then yields 28 different event sequences for $A \vec{\wedge} B$.

$$\begin{aligned}
(X_1 \vee X_5 \vee X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} B &= \dots = \\
&= [(\neg X_1 \neg X_{13}) \wedge ((X_5 \vee X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10})] \vee && \langle \text{from ES1} \rangle \\
&\vee [(\neg X_5 \neg X_{15}) \wedge ((X_1 \vee X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10})] \vee && \langle \text{from ES2} \rangle \\
&\vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_1 X_5))] \vee && \langle \text{from ES3} \rangle \\
&\vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_1 \vec{\wedge} X_5)] \vee && \langle \text{from ES3} \rangle
\end{aligned}$$

$$\begin{aligned}
& \vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_1)] \vee && \langle \text{from ES3} \rangle \\
& \vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_1 X_{15}))] \vee && \langle \text{from ES4} \rangle \\
& \vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (X_1 \vec{\wedge} X_{15})] \vee && \langle \text{from ES4} \rangle \\
& \vee [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_5 X_{13}))] \vee && \langle \text{from ES5} \rangle \\
& \vee [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_{13})] \vee && \langle \text{from ES5} \rangle \\
& \vee [(\neg X_1 \neg X_5 \neg X_{10}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_{13} X_{15}))] \vee && \langle \text{from ES6} \rangle \\
& \vee [(\neg X_1 \neg X_{10}) \wedge ((X_5 \wedge X_{15}) \vec{\wedge} X_{13})] \vee && \langle \text{from ES7} \rangle \\
& \vee [(\neg X_5 \neg X_{10}) \wedge ((X_1 \wedge X_{13}) \vec{\wedge} X_{15})] \vee && \langle \text{from ES8} \rangle \\
& \vee [(\neg X_{10} \neg X_{15}) \wedge ((X_1 \wedge X_{13}) \vec{\wedge} X_5)] \vee && \langle \text{from ES9} \rangle \\
& \vee [(\neg X_{10} \neg X_{13}) \wedge ((X_5 \wedge X_{15}) \vec{\wedge} X_1)] . && \langle \text{from ES10} \rangle \quad (7.27)
\end{aligned}$$

Thus, $A \vec{\wedge} B$ alone provides 12 event sequences of rank two and 16 event sequences of rank three.

Simplification of $(A \vec{\wedge} B) \wedge C$:

Using the TFTA's temporal logic, the meshing between event B and C in the sixth and last sub-expression of (7.3) may be solved.

According to (7.1) B and C are given as

$$B = X_{10} \vee [(X_1 \vee X_{13}) \wedge (X_5 \vee X_{15})] \quad \text{and} \quad (7.28)$$

$$C = X_{28} \vee [(X_1 \vee X_{13}) \wedge (X_5 \vee X_{15})] \vee [(X_{27} \vee X_{18}) \vec{\wedge} X_{10}] . \quad (7.29)$$

Further substitution with

$$D = (X_1 \vee X_{13}) \wedge (X_5 \vee X_{15}) = X_1 X_5 \vee X_1 X_{15} \vee X_5 X_{13} \vee X_{13} X_{15} \quad (7.30)$$

uncovers the relationship between B and C :

$$B = X_{10} \vee D \quad \text{and} \quad (7.31)$$

$$C = X_{28} \vee D \vee ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) . \quad (7.32)$$

Applying (7.31) and (7.32) provides

$$\begin{aligned}
(A \vec{\wedge} B) \wedge C &= (A \vec{\wedge} B) \wedge (X_{28} \vee D \vee ((X_{27} \vee X_{18}) \vec{\wedge} X_{10})) = \\
&= [(A \vec{\wedge} B) \wedge X_{28}] \vee [(A \vec{\wedge} B) \wedge D] \vee [(A \vec{\wedge} B) \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10})] . \quad (7.33)
\end{aligned}$$

The first expression yields (without SAND)

$$(A \vec{\wedge} B) \wedge X_{28} = [A \vec{\wedge} B \vec{\wedge} X_{28}] \vee [(A \wedge X_{28}) \vec{\wedge} B] . \quad (7.34)$$

The TDNF of $(A \vec{\wedge} B) \wedge X_{28}$ consists of 56 MCSS in total. $[A \vec{\wedge} B \vec{\wedge} X_{28}]$ provides 28 MCSS, each similar to those in (7.27) but extended by an additional X_{28} . $[(A \wedge X_{28}) \vec{\wedge} B]$ also provides 28 MCSS similar to those in (7.27). Instead of A the expression $A \wedge X_{28}$ is used, respectively. 24 of the MCSS are of rank three and 32 of the MCSS are of rank four.

The second expression in (7.33) provides (without SAND)

$$\begin{aligned}
(A \vec{\wedge} B) \wedge D &= (A \vec{\wedge} (X_{10} \vee D)) \wedge D = \dots = \\
&= [\neg X_{10} \wedge (A \vec{\wedge} D)] \vee [A \vec{\wedge} X_{10} \vec{\wedge} D] . \quad (7.35)
\end{aligned}$$

$[\neg X_{10} \wedge (A \vec{\wedge} D)]$ provides 20 MCSS similar to those in (7.27). As D does not include event X_{10} (other than B), the first eight event sequences may be dropped, i.e. the first two rows in (7.27). In the other rows the $\neg X_{10}$ are also dropped. Therefore,

$$A \vec{\wedge} D = A \vec{\wedge} B \Big|_{X_{10} = \text{False}} . \quad (7.36)$$

For expression $[\neg X_{10} \wedge (A \vec{\wedge} D)]$ only four MCSS of rank two and 16 MCSS of rank three remain, see (7.37).

$$\begin{aligned} \neg X_{10} \wedge (A \vec{\wedge} D) = & [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_1 X_5))] \vee \\ & \vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_1 \vec{\wedge} X_5)] \vee \\ & \vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_1)] \vee \\ & \vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_1 X_{15}))] \vee \\ & \vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (X_1 \vec{\wedge} X_{15})] \vee \\ & \vee [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_5 X_{13}))] \vee \\ & \vee [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_{13})] \vee \\ & \vee [(\neg X_1 \neg X_5 \neg X_{10}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_{13} X_{15}))] \vee \\ & \vee [(\neg X_1 \neg X_{10}) \wedge ((X_5 \wedge X_{15}) \vec{\wedge} X_{13})] \vee \\ & \vee [(\neg X_5 \neg X_{10}) \wedge ((X_1 \wedge X_{13}) \vec{\wedge} X_{15})] \vee \\ & \vee [(\neg X_{10} \neg X_{15}) \wedge ((X_1 \wedge X_{13}) \vec{\wedge} X_5)] \vee \\ & \vee [(\neg X_{10} \neg X_{13}) \wedge ((X_5 \wedge X_{15}) \vec{\wedge} X_1)] . \end{aligned} \quad (7.37)$$

The expression in (7.37) provides 20 MCSS. Because of the additional X_{10} , four of those MCSS are of rank three and 16 are of rank four, see (7.38).

$$\begin{aligned} \neg X_{10} \wedge (A \vec{\wedge} D) = & [(\neg X_{13} \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10} \vec{\wedge} (X_1 X_5))] \vee \\ & \vee [(\neg X_{13} \neg X_{15}) \wedge (X_1 \vec{\wedge} X_{10} \vec{\wedge} X_5)] \vee \\ & \vee [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_{10} \vec{\wedge} X_1)] \vee \\ & \vee [(\neg X_5 \neg X_{13}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10} \vec{\wedge} (X_1 X_{15}))] \vee \\ & \vee [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (X_1 \vec{\wedge} X_{10} \vec{\wedge} X_{15})] \vee \\ & \vee [(\neg X_1 \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10} \vec{\wedge} (X_5 X_{13}))] \vee \\ & \vee [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_{10} \vec{\wedge} X_{13})] \vee \\ & \vee [(\neg X_1 \neg X_5) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10} \vec{\wedge} (X_{13} X_{15}))] \vee \\ & \vee [\neg X_1 \wedge ((X_5 \wedge X_{15}) \vec{\wedge} X_{10} \vec{\wedge} X_{13})] \vee \\ & \vee [\neg X_5 \wedge ((X_1 \wedge X_{13}) \vec{\wedge} X_{10} \vec{\wedge} X_{15})] \vee \\ & \vee [\neg X_{15} \wedge ((X_1 \wedge X_{13}) \vec{\wedge} X_{10} \vec{\wedge} X_5)] \vee \\ & \vee [\neg X_{13} \wedge ((X_5 \wedge X_{15}) \vec{\wedge} X_{10} \vec{\wedge} X_1)] . \end{aligned} \quad (7.38)$$

The transformation of the third expression $[(A \vec{\wedge} B) \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10})]$, see (7.33), is best demonstrated separately for each of the event sequences $\langle \text{ES1} \rangle$ to $\langle \text{ES10} \rangle$ in (7.26).

$\langle \text{ES1} \rangle$ and $\langle \text{ES2} \rangle$ differ in the relevant events; therefore

$$\begin{aligned} \langle \text{ES1} \rangle : & [(\neg X_1 \neg X_{13}) \wedge ((X_5 \vee X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} X_{10})] \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) = \\ & = (\neg X_1 \neg X_{13}) \wedge ((X_5 X_{18}) \vee (X_{20} X_{18}) \vee (X_{22} X_{18}) \vee (X_{24} X_{18}) \vee \end{aligned}$$

$$\begin{aligned}
& \vee (X_5 X_{27}) \vee (X_{20} X_{27}) \vee (X_{22} X_{27}) \vee (X_{24} X_{27}) \vec{\wedge} X_{10} \quad \text{and} \\
\langle \text{ES2} \rangle : & \quad (\neg X_5 \neg X_{15}) \wedge ((X_1 X_{18}) \vee (X_{20} X_{18}) \vee (X_{22} X_{18}) \vee (X_{24} X_{18}) \vee \\
& \quad \vee (X_1 X_{27}) \vee (X_{20} X_{27}) \vee (X_{22} X_{27}) \vee (X_{24} X_{27})) \vec{\wedge} X_{10} . \tag{7.39}
\end{aligned}$$

The first part of $\langle \text{ES3} \rangle$ provides

$$\langle \text{ES3} \rangle : \quad [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge ((X_{20} \vee X_{22} \vee X_{24}) \vec{\wedge} (X_1 X_5))] \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) . \tag{7.40}$$

Further simplification yields only event sequences of rank five and higher. These are not further considered, as they are far more improbable than the other MCSS, which contribute significantly more. Such a reduction of the necessary effort is state of the art in conventional FTA, too. The same is true for the simplification of the first part of $\langle \text{ES4} \rangle$ and $\langle \text{ES5} \rangle$, as well as for all of $\langle \text{ES6} \rangle$ to $\langle \text{ES10} \rangle$.

The second part of $\langle \text{ES3} \rangle$ provides four MCSS of rank four:

$$\begin{aligned}
\langle \text{ES3} \rangle : & \quad [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_1 \vec{\wedge} X_5)] \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) = \\
& = \quad [(\neg X_{13} \neg X_{15} \neg X_{27}) \wedge (X_1 \vec{\wedge} X_5 \vec{\wedge} X_{18} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_{13} \neg X_{15} \neg X_{18}) \wedge (X_1 \vec{\wedge} X_5 \vec{\wedge} X_{27} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_{13} \neg X_{15} \neg X_{27}) \wedge ((X_1 X_{18}) \vec{\wedge} X_5 \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_{13} \neg X_{15} \neg X_{18}) \wedge ((X_1 X_{27}) \vec{\wedge} X_5 \vec{\wedge} X_{10})] . \tag{7.41}
\end{aligned}$$

Analogously, the third part of $\langle \text{ES3} \rangle$ and the second parts of $\langle \text{ES4} \rangle$ and $\langle \text{ES5} \rangle$ also provide four MCSS of rank four, respectively:

$$\begin{aligned}
\langle \text{ES3} \rangle : & \quad [(\neg X_{10} \neg X_{13} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_1)] \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) = \\
& = \quad [(\neg X_{13} \neg X_{15} \neg X_{27}) \wedge (X_5 \vec{\wedge} X_1 \vec{\wedge} X_{18} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_{13} \neg X_{15} \neg X_{18}) \wedge (X_5 \vec{\wedge} X_1 \vec{\wedge} X_{27} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_{13} \neg X_{15} \neg X_{27}) \wedge ((X_5 X_{18}) \vec{\wedge} X_1 \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_{13} \neg X_{15} \neg X_{18}) \wedge ((X_5 X_{27}) \vec{\wedge} X_1 \vec{\wedge} X_{10})] . \tag{7.42}
\end{aligned}$$

$$\begin{aligned}
\langle \text{ES4} \rangle : & \quad [(\neg X_5 \neg X_{10} \neg X_{13}) \wedge (X_1 \vec{\wedge} X_{15})] \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) = \\
& = \quad [(\neg X_5 \neg X_{13} \neg X_{27}) \wedge (X_1 \vec{\wedge} X_{15} \vec{\wedge} X_{18} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_5 \neg X_{13} \neg X_{18}) \wedge (X_1 \vec{\wedge} X_{15} \vec{\wedge} X_{27} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_5 \neg X_{13} \neg X_{27}) \wedge ((X_1 X_{18}) \vec{\wedge} X_{15} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_5 \neg X_{13} \neg X_{18}) \wedge ((X_1 X_{27}) \vec{\wedge} X_{15} \vec{\wedge} X_{10})] . \tag{7.43}
\end{aligned}$$

$$\begin{aligned}
\langle \text{ES5} \rangle : & \quad [(\neg X_1 \neg X_{10} \neg X_{15}) \wedge (X_5 \vec{\wedge} X_{13})] \wedge ((X_{27} \vee X_{18}) \vec{\wedge} X_{10}) = \\
& = \quad [(\neg X_1 \neg X_{15} \neg X_{27}) \wedge (X_5 \vec{\wedge} X_{13} \vec{\wedge} X_{18} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_1 \neg X_{15} \neg X_{18}) \wedge (X_5 \vec{\wedge} X_{13} \vec{\wedge} X_{27} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_1 \neg X_{15} \neg X_{27}) \wedge ((X_5 X_{18}) \vec{\wedge} X_{13} \vec{\wedge} X_{10})] \vee \\
& \quad \vee [(\neg X_1 \neg X_{15} \neg X_{18}) \wedge ((X_5 X_{27}) \vec{\wedge} X_{13} \vec{\wedge} X_{10})] . \tag{7.44}
\end{aligned}$$

7.3.3 Analysis of the MCSS

The MCSS of the temporal failure function ϖ are derived from the event sequences of the sub-expressions in (7.3), which are not necessarily already MCSS, i.e. there could be intersections

and overlaps between these individual expressions. In general, MCSS of smaller rank are those with higher importance. Therefore, the following discussion focusses on MCSS of rank two and three.

Event Sequences of the Resulting Expressions

The expressions' event sequences of rank two and three are listed in table 7.4. They are derived from table 7.3 on page 92 as well as the equations (7.34), (7.37), (7.38), and (7.39).

(extended) event sequences of rank two:		
1: $X_1 \vec{\wedge} X_5$	3: $X_1 \vec{\wedge} X_{15}$	5: $X_{28} \wedge X_{38}$
2: $X_5 \vec{\wedge} X_1$	4: $X_5 \vec{\wedge} X_{13}$	
(extended) event sequences of rank three:		
1: $X_{28} \wedge X_{30} \wedge X_{32}$	25: $(X_5 \wedge X_{28}) \vec{\wedge} X_{10}$	49: $X_1 \vec{\wedge} X_{10} \vec{\wedge} X_5$
2: $X_{28} \wedge X_{30} \wedge X_{36}$	26: $(X_{20} \wedge X_{28}) \vec{\wedge} X_{10}$	50: $X_5 \vec{\wedge} X_{10} \vec{\wedge} X_1$
3: $X_{28} \wedge X_{32} \wedge X_{34}$	27: $(X_{22} \wedge X_{28}) \vec{\wedge} X_{10}$	51: $X_1 \vec{\wedge} X_{10} \vec{\wedge} X_{15}$
4: $X_{28} \wedge X_{34} \wedge X_{36}$	28: $(X_{24} \wedge X_{28}) \vec{\wedge} X_{10}$	52: $X_5 \vec{\wedge} X_{10} \vec{\wedge} X_{13}$
5: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} X_{38}$	29: $(X_1 \wedge X_{28}) \vec{\wedge} X_{10}$	53: $(X_5 \wedge X_{18}) \vec{\wedge} X_{10}$
6: $(X_{18} \wedge X_{38}) \vec{\wedge} X_{10}$	30: $(X_{20} \wedge X_{28}) \vec{\wedge} X_{10}$	54: $(X_{18} \wedge X_{20}) \vec{\wedge} X_{10}$
7: $X_1 \wedge X_5 \wedge X_{38}$	31: $(X_{22} \wedge X_{28}) \vec{\wedge} X_{10}$	55: $(X_{18} \wedge X_{22}) \vec{\wedge} X_{10}$
8: $X_1 \wedge X_{15} \wedge X_{38} \diamond$	32: $(X_{24} \wedge X_{28}) \vec{\wedge} X_{10}$	56: $(X_{18} \wedge X_{24}) \vec{\wedge} X_{10}$
9: $X_5 \wedge X_{13} \wedge X_{38} \diamond$	33: $(X_1 \wedge X_{28}) \vec{\wedge} X_5$	57: $(X_1 \wedge X_{18}) \vec{\wedge} X_{10}$
10: $X_{13} \wedge X_{15} \wedge X_{38}$	34: $(X_5 \wedge X_{28}) \vec{\wedge} X_1$	58: $(X_{18} \wedge X_{20}) \vec{\wedge} X_{10}$
11: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} X_{38}$	35: $(X_1 \wedge X_{28}) \vec{\wedge} X_{15}$	59: $(X_{18} \wedge X_{22}) \vec{\wedge} X_{10}$
12: $(X_{27} \wedge X_{38}) \vec{\wedge} X_{10}$	36: $(X_5 \wedge X_{28}) \vec{\wedge} X_{13}$	60: $(X_{18} \wedge X_{24}) \vec{\wedge} X_{10}$
13: $X_5 \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	37: $X_{20} \vec{\wedge} (X_1 \wedge X_5)$	61: $(X_5 \wedge X_{27}) \vec{\wedge} X_{10}$
14: $X_{20} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	38: $X_{22} \vec{\wedge} (X_1 \wedge X_5)$	62: $(X_{20} \wedge X_{27}) \vec{\wedge} X_{10}$
15: $X_{22} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	39: $X_{24} \vec{\wedge} (X_1 \wedge X_5)$	63: $(X_{22} \wedge X_{27}) \vec{\wedge} X_{10}$
16: $X_{24} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	40: $X_{20} \vec{\wedge} (X_1 \wedge X_{15}) \diamond$	64: $(X_{24} \wedge X_{27}) \vec{\wedge} X_{10}$
17: $X_1 \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	41: $X_{22} \vec{\wedge} (X_1 \wedge X_{15}) \diamond$	65: $(X_1 \wedge X_{27}) \vec{\wedge} X_{10}$
18: $X_{20} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	42: $X_{24} \vec{\wedge} (X_1 \wedge X_{15}) \diamond$	66: $(X_{20} \wedge X_{27}) \vec{\wedge} X_{10}$
19: $X_{22} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	43: $X_{20} \vec{\wedge} (X_5 \wedge X_{13}) \diamond$	67: $(X_{22} \wedge X_{27}) \vec{\wedge} X_{10}$
20: $X_{24} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	44: $X_{22} \vec{\wedge} (X_5 \wedge X_{13}) \diamond$	68: $(X_{24} \wedge X_{27}) \vec{\wedge} X_{10}$
21: $X_1 \wedge X_5 \wedge X_{28}$	45: $X_{24} \vec{\wedge} (X_5 \wedge X_{13}) \diamond$	69: $(X_5 \wedge X_{15}) \vec{\wedge} X_{13}$
22: $X_5 \vec{\wedge} X_1 \vec{\wedge} X_{28}$	46: $X_{20} \vec{\wedge} (X_{13} \wedge X_{15})$	70: $(X_1 \wedge X_{13}) \vec{\wedge} X_{15}$
23: $X_1 \vec{\wedge} X_{15} \vec{\wedge} X_{28}$	47: $X_{22} \vec{\wedge} (X_{13} \wedge X_{15})$	71: $(X_1 \wedge X_{13}) \vec{\wedge} X_5$
24: $X_5 \vec{\wedge} X_{13} \vec{\wedge} X_{28}$	48: $X_{24} \vec{\wedge} (X_{13} \wedge X_{15})$	72: $(X_5 \wedge X_{15}) \vec{\wedge} X_1$

Table 7.4: Event sequences of rank two and three. Event sequences which are included more than once have an “underwave”, non-minimal event sequences are ~~striked through~~, “partly” non-minimal event sequences are marked with \diamond .

Minimal Form and MCSS of the Failure Funktion

A total of 12 of the 77 event sequences in table 7.4 are included at least twice and may be omitted using the law of idempotency. A further 20 event sequences are non-minimal and also omitted. The extended event sequences number 8 and 9 and 40 to 45, i.e.

$$\begin{aligned}
& [X_1 \wedge X_{15} \wedge X_{38}], [X_5 \wedge X_{13} \wedge X_{38}], \\
& [X_{20} \vec{\wedge} (X_1 \wedge X_{15})], [X_{22} \vec{\wedge} (X_1 \wedge X_{15})], [X_{24} \vec{\wedge} (X_1 \wedge X_{15})], \\
& [X_{20} \vec{\wedge} (X_5 \wedge X_{13})], [X_{22} \vec{\wedge} (X_5 \wedge X_{13})], [X_{24} \vec{\wedge} (X_5 \wedge X_{13})],
\end{aligned} \tag{7.45}$$

are “partly” non-minimal with respect to the MCSS of rank two, i.e.

$$[X_1 \vec{\wedge} X_{15}] \quad \text{and} \quad [X_5 \vec{\wedge} X_{13}] . \quad (7.46)$$

Therefore, it is necessary to break up the extended event sequences in order to separate their minimal and non-minimal parts.

For example, the event sequence $X_{20} \vec{\wedge} (X_1 \wedge X_{15})$ provides (without SAND) two non-extended (normal) event sequences, i.e.

$$X_{20} \vec{\wedge} (X_1 \wedge X_{15}) = [(X_1 X_{20}) \vec{\wedge} X_{15}] \vee [(X_{15} X_{20}) \vec{\wedge} X_1] , \quad (7.47)$$

where the first is non-minimal with respect to $X_1 \vec{\wedge} X_{15}$.

In analogy to that,

$$X_1 \wedge X_{15} \wedge X_{38} = [(X_1 \vec{\wedge} X_{15}) \wedge X_{38}] \vee [(X_{15} \vec{\wedge} X_1) \wedge X_{38}] . \quad (7.48)$$

Only the second event sequence is minimal. It is first transformed into a TDNF, thus

$$(X_{15} \vec{\wedge} X_1) \wedge X_{38} = [X_{15} \vec{\wedge} X_1 \vec{\wedge} X_{38}] \vee [(X_{15} \wedge X_{38}) \vec{\wedge} X_1] . \quad (7.49)$$

Therefore, the two partly minimal event sequences number 8 and 9 provide four minimal MCSS.

Table 7.5 shows a cleaned up list, in which only MCSS of rank two and three of the failure function ϖ are shown.

(extended) MCSS of rank two:		
1: $X_1 \vec{\wedge} X_5$	3: $X_1 \vec{\wedge} X_{15}$	5: $X_{28} \wedge X_{38}$
2: $X_5 \vec{\wedge} X_1$	4: $X_5 \vec{\wedge} X_{13}$	
(extended) MCSS of rank three:		
1: $X_{28} \wedge X_{30} \wedge X_{32}$	15: $(X_5 \wedge X_{28}) \vec{\wedge} X_{10}$	29: $X_{20} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$
2: $X_{28} \wedge X_{30} \wedge X_{36}$	16: $(X_{20} \wedge X_{28}) \vec{\wedge} X_{10}$	30: $X_{22} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$
3: $X_{28} \wedge X_{32} \wedge X_{34}$	17: $(X_{22} \wedge X_{28}) \vec{\wedge} X_{10}$	31: $X_{24} \vec{\wedge} X_{10} \vec{\wedge} X_{28}$
4: $X_{28} \wedge X_{34} \wedge X_{36}$	18: $(X_{24} \wedge X_{28}) \vec{\wedge} X_{10}$	32: $X_1 \vec{\wedge} X_{10} \vec{\wedge} X_{28}$
5: $X_{18} \vec{\wedge} X_{10} \vec{\wedge} X_{38}$	19: $(X_1 \wedge X_{28}) \vec{\wedge} X_{10}$	33: $(X_5 \wedge X_{18}) \vec{\wedge} X_{10}$
6: $(X_{18} \wedge X_{38}) \vec{\wedge} X_{10}$	20: $(X_{15} \wedge X_{20}) \vec{\wedge} X_1$	34: $(X_{18} \wedge X_{20}) \vec{\wedge} X_{10}$
7: $X_{15} \vec{\wedge} X_1 \vec{\wedge} X_{38}$	21: $(X_{15} \wedge X_{22}) \vec{\wedge} X_1$	35: $(X_{18} \wedge X_{22}) \vec{\wedge} X_{10}$
8: $(X_{15} \wedge X_{38}) \vec{\wedge} X_1$	22: $(X_{15} \wedge X_{24}) \vec{\wedge} X_1$	36: $(X_{18} \wedge X_{24}) \vec{\wedge} X_{10}$
9: $X_{13} \vec{\wedge} X_5 \vec{\wedge} X_{38}$	23: $(X_{13} \wedge X_{20}) \vec{\wedge} X_5$	37: $(X_1 \wedge X_{18}) \vec{\wedge} X_{10}$
10: $(X_{13} \wedge X_{38}) \vec{\wedge} X_5$	24: $(X_{13} \wedge X_{22}) \vec{\wedge} X_5$	38: $(X_5 \wedge X_{27}) \vec{\wedge} X_{10}$
11: $X_{13} \wedge X_{15} \wedge X_{38}$	25: $(X_{13} \wedge X_{24}) \vec{\wedge} X_5$	39: $(X_{20} \wedge X_{27}) \vec{\wedge} X_{10}$
12: $X_{27} \vec{\wedge} X_{10} \vec{\wedge} X_{38}$	26: $X_{20} \vec{\wedge} (X_{13} \wedge X_{15})$	40: $(X_{22} \wedge X_{27}) \vec{\wedge} X_{10}$
13: $(X_{27} \wedge X_{38}) \vec{\wedge} X_{10}$	27: $X_{22} \vec{\wedge} (X_{13} \wedge X_{15})$	41: $(X_{24} \wedge X_{27}) \vec{\wedge} X_{10}$
14: $X_5 \vec{\wedge} X_{10} \vec{\wedge} X_{28}$	28: $X_{24} \vec{\wedge} (X_{13} \wedge X_{15})$	42: $(X_1 \wedge X_{27}) \vec{\wedge} X_{10}$

Table 7.5: MCSS of rank two and three. This table is a version of table 7.4, but stripped of non-minimal event sequences and duplicates.

Results

The MCSS of the failure function are all of ranks two and higher. Therefore, no single failure within the system as modelled leads directly to an infraction of the safety goal. The example system thus satisfies the requirement of single-failure-resistance, as described in chapter 7.2.

The most important combinations of dangerous failures, that lead to an infraction of the safety goal, are MCSS of rank two and three. The five MCSS of rank two are

1. either failures of the two sensors following each other. In this case EN1 would be activated by the first sensor failure, and SAF would be activated by the second sensor failure. These two failures may occur in arbitrary sequence.
2. or one sensor failure in combination with a failure of μC . The sensor failure needs to occur before the failure of the μC , otherwise the sequence logic in L would not be activated.
3. or am failure of T3 in combination with a failure of L, which activates both power stages. These two failures may occur in arbitrary sequence.

MCSS of rank three are e.g.

1. a double failure of the high side and the low side of the driver in combination with an internal failure in T3. No sequence logic has to be respected here. Specifically, numbers 1 to 4 in table 7.5 are combinations of this type.
2. failures of the system ASIC in combination with failures of the μC and/or sensor failures. Specifically, numbers 20 to 28 in table 7.5 are combinations of this type.
3. a failure of the watchdog or of the emergency switch in combination with an ASIC failure, where both occur before an additional failure of the μC , see, for instance, numbers 34 to 36 and 39 to 41 in table 7.5.
4. a failure in one of the sensors in combination with a failure of the watchdog or the emergency switch, where both occur before an additional failure of the μC , see, for instance, numbers 33, 37, 38, and 42 in table 7.5.

7.4 Probabilistic Analysis of the TOP Failure Parameters

The qualitative analysis of the temporal fault tree is used as evidence that the system stays below the threshold for failure rates as required by ISO 26262 for ASIL D systems. This threshold is given as $\leq 1 \cdot 10^{-8} \frac{1}{h}$ for any operating hour during the whole mission time.

In order to do so, it has to be demonstrated, that the failure rate of the TOP event λ_{TOP} stays below this threshold.

Because of $f_{TOP}(T_M) \approx \lambda_{TOP}$, see (5.59), it is sufficient to use the TOP event's failure frequency as a good approximation.

Furthermore, an iterative multi-step approach is chosen, that reduces effort and is used in similar fashion in many real world FTA analyses. First, an approximation with conservative estimations of the failure rates is used that allows for a first overview.

The evidence is sufficiently produced if, using this approach, the thresholds, as required by the safety standard, are not exceeded. If this can not be shown, the next step is to determine the failure rates more exactly and/or use exact calculations instead of approximations – and to possibly restrict the further analysis to the most important contributors as identified in the first step's overview. The termination condition for these steps is that the thresholds, as required by the safety standard, are no longer exceeded.

Because of this, in the following discussion the MCSS are not transformed into a mutually exclusive (disjoint) form. Instead, the approximation approach from chapter 5.5 is used. This corresponds to the bottom most path in figure 3.1 on page 18.

Quantification of the failure function ϖ is carried out using its MCSS from table 7.5. All basic events are allocated the same failure rate of $\lambda = 10^{-6} \frac{1}{h}$.

Table 7.6 shows failure probabilities and failure frequencies according to (5.74) and (5.75) for each MCSS from table 7.5. The mission time is given as $T_M = 1000\text{h}$.

MCSS of rank two:		
1: $F=5 \cdot 10^{-7}; f=1 \cdot 10^{-9} \frac{1}{\text{h}}$	3: $F=5 \cdot 10^{-7}; f=1 \cdot 10^{-9} \frac{1}{\text{h}}$	5: $F=1 \cdot 10^{-6}; f=2 \cdot 10^{-9} \frac{1}{\text{h}}$
2: $F=5 \cdot 10^{-7}; f=1 \cdot 10^{-9} \frac{1}{\text{h}}$	4: $F=5 \cdot 10^{-7}; f=1 \cdot 10^{-9} \frac{1}{\text{h}}$	
MCSS of rank three:		
1: $F=1 \cdot 10^{-9}; f=3 \cdot 10^{-12} \frac{1}{\text{h}}$	15: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	29: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
2: $F=1 \cdot 10^{-9}; f=3 \cdot 10^{-12} \frac{1}{\text{h}}$	16: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	30: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
3: $F=1 \cdot 10^{-9}; f=3 \cdot 10^{-12} \frac{1}{\text{h}}$	17: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	31: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
4: $F=1 \cdot 10^{-9}; f=3 \cdot 10^{-12} \frac{1}{\text{h}}$	18: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	32: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
5: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	19: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	33: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
6: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	20: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	34: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
7: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	21: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	35: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
8: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	22: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	36: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
9: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	23: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	37: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
10: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	24: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	38: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
11: $F=1 \cdot 10^{-9}; f=3 \cdot 10^{-12} \frac{1}{\text{h}}$	25: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	39: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
12: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	26: $F=\frac{2}{3} \cdot 10^{-9}; f=2 \cdot 10^{-12} \frac{1}{\text{h}}$	40: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
13: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	27: $F=\frac{2}{3} \cdot 10^{-9}; f=2 \cdot 10^{-12} \frac{1}{\text{h}}$	41: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$
14: $F=\frac{1}{6} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$	28: $F=\frac{2}{3} \cdot 10^{-9}; f=2 \cdot 10^{-12} \frac{1}{\text{h}}$	42: $F=\frac{1}{3} \cdot 10^{-9}; f=1 \cdot 10^{-12} \frac{1}{\text{h}}$

Table 7.6: Failure probabilities and failure frequencies according to (5.74) and (5.75) for each MCSS from table 7.5.

Failure characteristics at TOP event level are then calculated using (5.55) and (5.56), respectively, as sum of the individual MCSS' contributions. Using the values from table 7.6 yields

$$F_{TOP}(T_M) \approx 3,017 \cdot 10^{-6} \quad \text{and} \quad (7.50)$$

$$f_{TOP}(T_M) \approx 6,055 \cdot 10^{-9} \frac{1}{\text{h}}. \quad (7.51)$$

This first approximation already provides the evidence for meeting the ISO 26262 standard's requirements for ASIL D; the TOP event's failure frequency in (7.51) stays well below the threshold of $1 \cdot 10^{-8} \frac{1}{\text{h}}$.

Remark: With conventional FTA the PAND gate would have to be replaced by normal AND gates. This would affect the failure frequencies of minimal cutsets of rank two the most. These minimal cutsets would be the same as the MCSS of rank two, only using AND operators instead of the PANDs. Accordingly, in an Boolean FTA the TOP event's failure frequency would nearly double compared to the TFTA's result, yielding $> 1 \cdot 10^{-8} \frac{1}{\text{h}}$ and, thus, exceeding the threshold limit.

7.5 Discussion

The analysis of this real world example system in chapter 7.1 demonstrates that the TFTA method is not limited to modelling only very small examples. Chapter 7 thereby extends the theoretical discussions on the TFTA approach in chapters 4 and 5, as well as the statements on basic application of the TFTA in chapter 6.

The analogy to the conventional FTA is shown during the creation of the temporal fault tree in figures 7.2 to 7.4. In this process no additional effort is necessary in comparison to the Boolean FTA apart from choosing temporal fault tree gates.

In this temporal fault tree there are several meshings of basic events as well as of whole subtrees. For instance, events beneath “ μ C signal failure” are found beneath a temporal gate (“L commanded failure”). The same events are also found in the purely Boolean part of the fault tree below of “commanded failure T3”. Additionally, the basic event “10 – μ C generic failure activates SAF” is found in different and otherwise separated subtrees beneath different PAND gates.

Using such meshing in e.g. an DFT approach would dramatically increase the effort; the necessary separation into different dynamic and non-dynamic modules would require that almost the whole fault tree had to be modelled as a dynamic module, i.e. in case of the DFT it had to be modelled using markov methods.

The detailed qualitative analysis of the temporal fault tree in chapter 7.3 demonstrates that the TFTA is able to solve these meshings by use of its temporal transformation laws.

On the one hand it is true that the calculatory effort for these transformations increases rapidly, specifically because of the temporal distributive laws. On the other hand, the required calculations are mostly limited to string-manipulations. As a general rule, these are less costly than solving exponentially growing markov models or simulating big petri nets, as necessary for the other methods.

The analysis of the MCSS in chapter 7.3.3 is, then, very similar to the Boolean FTA. Among others, it is demonstrated that the TFTA is well suited for real qualitative analysis. As described in chapter 6, this is one of the main advantages of the TFTA.

The probabilistic quantification, as demonstrated in chapter 7.4, is based on a step-by-step approach, as is best praxis. This allows adjusting modelling precision to the issue at hand – which implies adjustable effort –, as well as concentrating all resources on the most important contributors. Both is not possible to the same extend when using the DFT.

8 Summary and Outlook

I want electricity to become so cheap
that only the rich can afford candles.

(Thomas Alva Edison)

The new approach to temporal fault tree analysis presented in this thesis is called TFTA; it extends the Boolean FTA in order to include event sequences. In comparison to the conventional FTA this allows a more realistic model of the failure behaviour of complex and dynamic systems.

The new TFTA uses a new temporal logic described in this thesis. With this logic it differs significantly from most existing approaches with similar aims. These transform the FTA model completely or partially into a state based model; temporal effects are then handled in the state space, and the results are then transferred back into the fault tree. TFTA contrasts with such state based methods in that

- it uses an extension to Boolean algebra and logic,
- its notation, terms, and its workflow and work products are taken from the conventional FTA,
- it allows qualitative as well as probabilistic analyses and calculations including event sequence information.

In comparison to other known approaches that also use a “temporal logic” to include temporal information into the fault tree the TFTA is significantly leaner.

Specifically, TFTA is not another attempt to create a formal FTA logic for modelling of software systems. Instead, TFTA emphasises practise-oriented characteristics like intuitive applicability, readability, comprehensible logic expressions and results, transferability of real world failure effects into the model, and scalability.

The temporal logic of the TFTA uses the Boolean operations of conjunction, disjunction, and negation. Additionally, two new temporal operations (PAND and SAND) represent two “special conjunctions” that describe event sequences and simultaneous events, respectively.

Using the well known Boolean algebra and a set of new temporal transformation laws, it is possible to transform complex temporal expressions into their temporal disjunctive normal form (TDNF) which consists of separate event sequences. In analogy to the Boolean fault tree cutsets these event sequences are reduced to a minimal form, the so-called minimal cutset sequences (MCSS).

Then, MCSS are made mutually exclusive (i.e. disjoint). This disjoint form is especially well suited for direct quantification and makes probabilistic analysis possible.

Other than conventional FTA, probabilistic TFTA allows to calculate reliability characteristics like failure probability, failure frequency, and failure rate of a fault tree TOP event with consideration of event sequence information, and without the need to change into the state space.

Evaluation of this Thesis

Originally, the development of an own temporal logic aimed primarily at solving some of the problems that arise with the known dynamic extensions of the FTA which are based on markov methods. The DFT method [37] is a well known representative of such dynamic extensions, and thus it is an obvious choice to compare what this thesis achieved with the DFT method.

With regard to the calculatory effort, the consideration of event sequences always implies additional cost when compared to the Boolean FTA. This is true for state based extensions, as well as for extended logics covering temporal effects. This additional cost is a concern, even more so, as the determination of disjoint minimal cutsets in Boolean FTA already carries exponentially growing complexity. On the other hand, the TFTA method does not aim at solving this.

Some of the TFTA's problems are fundamentally connected to the kind of temporal logic that is used. Event sequence statements only cover the points in time at which events occur. Therefore, "time-limited" failure events, i.e. events with a defined time span of being *True*, can not be represented by PAND and SAND. Instead, such effects need to be represented by conventional AND gates. This, however, is no deterioration in comparison to the DFT method. The markov chains that the DFT uses are also only able of capturing state transitions resulting from "initiating" failure events; it is not able to capture "time-limited" failure events. The DFT only hides this shortcoming better, because of the necessary modularization and because meshing is impossible.

One major shortcoming of the DFT is modularization. In some cases, it makes it impossible to mesh events beyond single dynamic fault tree gates logically correctly. Compared to that, TFTA allows for such meshing. It, thus, is possible to consider more event sequence effects.

Another major shortcoming of the DFT concerns qualitative evaluation of minimal cutsets. The transformation into the state space either forces the use of "meta events" in addition to basic events; these meta event represent complete markov models. As an alternative, qualitative analysis is restricted to not include event sequence information. Compared to that, the (extended) event sequences in TFTA show exact event sequence information of all basic events that contribute to the TOP failure. As such, the TFTA permits more meaningful and efficient qualitative analyses than the DFT.

Both, the TFTA as well as the DFT allow for probabilistic evaluation of the TOP event's failure rate and failure probability. On the one hand, with this quantification it is possible to determine the precise TOP event's failure characteristics at comparably high calculatory costs. On the other hand, an approximation for the TFTA is provided, which reduces the necessary effort significantly.

Three more arguments support the TFTA with regard to calculatory costs: first, the size of the differential equations system, necessary for solving the DFT, grows exponentially with the number of component failures that are within a dynamic module. Therefore, the overhead of TFTA (compared to Boolean FTA) is at least comparable with the DFT's overhead – and the TFTA provides more meaningful results, as discussed above. Second, calculations in the TFTA are mainly string-manipulations. These usually require less effort than solving exponentially growing state models. Third, the TFTA offers approximation methods, which provide a real

possibility to reduce overhead effectively, while accepting a certain degree of impreciseness; this may be used e.g. as a first step within a multi-step analysis.

Therefore, the TFTA is a capable replacement for the DFT's PAND gates, and furthermore provides some advantages methodology-wise, as well as for its useability.

Possible Further Research

During this theses several additional topics were discovered that could not be completely covered and solved within this work. For instance, SAND connections are defined as (structural) dependencies between failure events, and they are considered qualitatively, but they are not taken into account probabilistically. Because of the significance of dependent failures, which are sometime just called common cause failures (CCF), it seems promising to extend the TFTA method, as described in this thesis, by such dependencies. Furthermore, this thesis restricts itself to non-repairable failures. It seems possible that the TFTA's temporal logic, as well as the probabilistic aspects of the TFTA, may be extended to repairable failures. It could also be interesting to develop advanced methods to determine mutually exclusive (disjoint) expressions from a given TDNF. One possible way could be to follow segmentation-methods, like Abraham [80] or Heidtmann [81] proposed for Boolean algebra. Furthermore, it seems promising to investigate possible synergies between the TFTA logic and the BDD method in [86]. In general, there certainly is a demand for improved algorithms for using the TFTA in practise. In this regard, contributing to open source fault tree tools (like e.g. OpenFTA [87]) could be an interesting possibility.

9 Bibliography

- [1] K. D. Flörecke. Milliarden für mehr Sicherheit. *Automobilwoche 24*, page 14, 2004.
- [2] M. Meyer. *Methoden zur Analyse von Garantiedaten für Sicherheits- und Zuverlässigkeitsprognosen von Komponenten und Baugruppen im Kraftfahrzeug*. PhD thesis, Bergische Universität Wuppertal, Wuppertal, 2003.
- [3] ISO DIS 26262 Strassenfahrzeuge – Funktionssicherheit. Technical Committee ISO/TC 22, Road vehicles, Subcommittee SC 3, Electric and Electronic Equipment (in 10 parts), 2009.
- [4] IEC 61508 Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbar elektrischer Systeme (in 7 parts), 2002.
- [5] W. E. Veseley et al. *NUREG-0492 Fault tree handbook*. U.S. Nuclear Regulatory Commission, Washington, D.C., 1981.
- [6] IEC 61025 Edition 2.0 Fault tree analysis (FTA), 2006.
- [7] DIN 25424 Fehlerbaumanalyse (in 2 parts). Berlin, 1981 & 1990.
- [8] Winfrid G. Schneeweiss. *Die Fehlerbaum-Methode*. LiLoLe-Verlag, Hagen, 1999.
- [9] Verband der Automobilindustrie. *Fehlerbaumanalyse (Fault Tree Analysis FTA)*, volume 4 of *Sicherung der Qualität vor Serieneinsatz*. 2003.
- [10] Simon J. Schilling. Bedeutung und Modellierung abhängiger Ausfälle in automotiven E/E-Systemen. In *safetronic.2006*. Munich, 2006.
- [11] K.D. Heidtmann. Deterministic reliability-modeling of dynamic redundancy. *IEEE Transactions on Reliability*, 41(3):378–385, Sep 1992. ISSN 0018-9529. doi: 10.1109/24.159802.
- [12] R. Manian, J. Bechta Dugan, D. Coppit, and K.J. Sullivan. Combining various solution techniques for dynamic fault tree analysis of computer systems. In *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*, pages 21–28, Nov 1998. doi: 10.1109/HASE.1998.731591.
- [13] Jan Hauschild and Arno Meyna. Monte carlo techniques for modelling and analysing the reliability and safety of modern automotive applications. In Guedes, Soares, and Zio, editors, *Safety and Reliability for Managing Risk, ESREL 06*, London, 2006. Taylor and Francis Group.
- [14] Arno Meyna and Bernhard Pauli. *Taschenbuch der Zuverlässigkeits- und Sicherheitstechnik: quantitative Bewertungsverfahren*. Hanser, München, 2003.
- [15] DIN 40041: 1990-12: Zuverlässigkeit - Begriffe. Berlin, 1990.
- [16] Peter Bitter et al. *Technische Zuverlässigkeit - Problematik, math. Grundlagen, Untersuchungsmethoden, Anwendungen*. Springer, Berlin, 3rd edition, 1986.

- [17] Isograph Ltd. *AttackTree+ V1.0 Technical Specification*. Warrington, UK, 2005.
- [18] C. David Sulfredge, Robert L. Sanders, Douglas E. Peplow, and Robert H. Morris. Graphical Expert System for Analyzing Nuclear Facility Vulnerability. In *Transactions of Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*, Orlando, Florida, 2002.
- [19] Nathan O. Siu. Dynamic Approaches – Issues and Methods: An Overview. In [41], pages 3–7.
- [20] NRC. *Reactor safety study. An Assessment of accident risks in U.S. commercial nuclear power plants. WASH-1400. NUREG-75/014*. NRC, Washington, 1975.
- [21] Tunc Aldemir. Dynamic approaches – applications: An overview. In [41], pages 81–84.
- [22] Martin Wolterreck. *Dynamische Zuverlässigkeitsanalyse mit anlagenspezifischen Störfallsimulatoren*. PhD thesis, Technische Universität München, 2000.
- [23] L. Fahrmeir, H. Kaufmann, and F. Ost. *Stochastische Prozesse*. Hanser Verlag, München, 1982.
- [24] Jan Hauschild. *Beitrag zur Modellierung stochastischer Prozesse in der Sicherheits- und Zuverlässigkeitstechnik mittels Monte-Carlo-Simulation unter Berücksichtigung dynamischer Systemänderungen*. PhD thesis, Bergische Universität Wuppertal, 2007.
- [25] N. G. Leveson. *White Paper on Approaches to Safety Engineering*. Massachusetts, 2003.
- [26] H. A. Watson. *Launch Control Safety Study*. Bell Telephone Laboratories, Murray Hill, NJ, 1961.
- [27] S.V. Amari and J.B. Akers. Reliability analysis of large fault trees using the vesely failure rate. In *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, pages 391–396, Jan. 2004. doi: 10.1109/RAMS.2004.1285481.
- [28] T. Skorek. Determination of input uncertainties of uncertainty and sensitivity analyses. In *Probabilistic Safety Assessment and Management, PSAM 07 - ESREL 04*, Berlin, 2004. Springer.
- [29] Y. Dutuit and A. Rauzy. Efficient algorithms to assess component and gate importance in fault tree analysis. *Reliability Engineering and System Safety*, 72(2):213 – 222, 2001. ISSN 0951-8320. doi: 10.1016/S0951-8320(01)00004-7. URL <http://www.sciencedirect.com/science/article/B6V4T-42SGH2V-C/2/5eaa6fa2ef2eccab92a9f9e5e9e5f036>.
- [30] Martin Wolterreck and Ralph Vollmar. Reliability analysis of automotive systems: Quantification of data uncertainty. In *Probabilistic Safety Assessment and Management, PSAM 07 - ESREL 04*, Berlin, 2004. Springer.
- [31] W. E. Veseley et al. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, Washington, D.C., 2002.
- [32] P. Limbourg et al. Fault tree analysis in an early design stage using the dempster-shafer theory of evidence. In Aven and Vinnem, editors, *Risk, Reliability and Societal Safety, ESREL 07*, London, 2007. Taylor and Francis Group.

- [33] Wolfgang Weber, Heidemarie Tondok, and Michael Bachmayer. Enhancing Software Safety by Fault Trees: Experiences from Application to Flight Critical Software. In *SAFECOMP 2003*, pages 289–302, 2003.
- [34] Klaus Heidtmann. *Zuverlässigkeitsbewertung technischer Systeme*, volume 21 of *Teubner-Texte zur Informatik*. B. G. Teubner Verlagsgesellschaft, 1997.
- [35] J. D. Andrews. To Not or Not to Not! In *18th International System Safety Conference*, pages 267–275, 2000.
- [36] Heinz-Peter Gumm and Werner Poguntke. *Boolesche Algebra*. BI-Hochschultaschenbücher. Bibliogr. Inst., Mannheim, 1981.
- [37] J.B. Dugan, S.J. Bavuso, and M.A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, Sep 1992. ISSN 0018-9529. doi: 10.1109/24.159800.
- [38] Kevin J. Sullivan, Joanne Bechta Dugan, and David Coppit. The Galileo Fault Tree Analysis Tool. In *Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing*, pages 232–235, Madison, Wisconsin, 1999. IEEE.
- [39] Salvatore Distefano and Antonio Puliafito. Dynamic reliability block diagrams: Overview of a methodology. In Aven and Vinnem, editors, *Risk, Reliability and Societal Safety, ESREL 07*, London, 2007. Taylor and Francis Group.
- [40] Joanne Bechta Dugan, Dugan Kevin, David Coppit, and Kevin J. Sullivan. Developing a high-quality software tool for fault tree analysis. In *In Proceedings of the International Symposium on Software Reliability Engineering*, pages 49–59. IEEE, 1999.
- [41] Tunc Aldemir, Nathan O. Siu, Ali Mosleh, P. Carlo Cacciabue, and B. Gül Göktepe, editors. *Reliability and Safety Assessment of Dynamic Process Systems*, volume 120 of *NATO ASI Series F: Computer and System Sciences*. Springer Verlag, 1994.
- [42] Stefan Hirschberg and Michael Knochenhauer. Time dependencies in probabilistic safety assessment. In [41], pages 196–212.
- [43] Liudong Xing and J.B. Dugan. Analysis of generalized phased-mission system reliability, performance, and sensitivity. *IEEE Transactions on Reliability*, 51(2):199–211, Jun 2002. ISSN 0018-9529. doi: 10.1109/TR.2002.1011526.
- [44] C.J. Garrett, S.B. Guarro, and G.E. Apostolakis. The dynamic flowgraph methodology for assessing the dependability of embedded software systems. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5):824–840, May 1995. ISSN 0018-9472. doi: 10.1109/21.376495.
- [45] A. Kolaczowski et al. Human reliability analysis (hra) good practices. In *Probabilistic Safety Assessment and Management, PSAM 07 - ESREL 04*, Berlin, 2004. Springer.
- [46] Henrik Thane. Safe and reliable computer control systems: Concepts and methods. Technical report, 1996.
- [47] O. Coudert and J.C. Madre. Metaprime: an interactive fault-tree analyzer. *IEEE Transactions on Reliability*, 43(1):121–127, Mar 1994. ISSN 0018-9529. doi: 10.1109/24.285125.

- [48] W.S. Jung, S.H. Han, and J. Ha. Development of an efficient bdd algorithm to solve large fault trees. In *Probabilistic Safety Assessment and Management, PSAM 07 - ESREL 04*, Berlin, 2004. Springer.
- [49] Joanne Bechta Dugan, Bharath Venkataraman, and Rohit Gulati. DIFtree: A software package for the analysis of dynamic fault tree models. *Reliability and Maintainability Symposium*, pages 64–70, 1997.
- [50] Isograph Ltd. FaultTree+ V11.0. Warrington, UK. URL <http://www.isograph-software.com/ftpover.htm>.
- [51] ITEM Software Inc. ITEM ToolKit. Fareham, UK. URL <http://www.itemsoft.com/faulttree.shtml>.
- [52] Relex Software Corporation. *Relex Reliability Studio 2007*. 2007.
- [53] S. Montani, L. Portinale, A. Bobbio, and D. Codetta-Raiteri. Automatically translating dynamic fault trees into dynamic bayesian networks by means of a software tool. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 804–809, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2567-9.
- [54] Zhihua Tang and J.B. Dugan. Minimal cut set/sequence generation for dynamic fault trees. In *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, pages 207–213, 2004. doi: 10.1109/RAMS.2004.1285449.
- [55] Marco Bozzano and Adolfo Villafiorita. Integrating Fault Tree Analysis with Event Ordering Information. In *Safety and Reliability for Managing Risk, ESREL 03*, pages 247–254, 2003.
- [56] Marc Bouissou and Jean-Louis Bon. A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven markov processes. *Reliability Engineering and System Safety*, 82(2):149–163, 2003. ISSN 0951-8320. doi: 10.1016/S0951-8320(03)00143-1. URL <http://www.sciencedirect.com/science/article/B6V4T-49DFH1M-1/2/bd15510dc655e0bbc55f3e5758bdeb42>.
- [57] M. Bousissou. A Generalization of Dynamic Fault Trees through Boolean logic Driven Markov Processes (BDMP). In *ESREL 2007*, Stavanger (Norway), 2007.
- [58] A. Bobbio, G. Franceschinis, R. Gaeta, and L. Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level petri net semantics. *IEEE Transactions on Software Engineering*, 29(3):270–287, 2003. ISSN 0098-5589. doi: 10.1109/TSE.2003.1183940.
- [59] W. G. Schneeweiss. Advanced fault tree modeling. *Journal of Universal Computer Science*, 5(10):633–643, 1999.
- [60] Winfried G. Schneeweiss. *Petri Nets for Reliability Modeling*. LiLoLe-Verlag, 1999.
- [61] Bernhard Kaiser, Catharina Gramlich, and Marc Förster. State/event fault trees—a safety analysis model for software-controlled systems. *Reliability Engineering and System Safety*, 92(11):1521–1537, 2007. ISSN 0951-8320. doi: 10.1016/j.res.2006.10.010. URL <http://www.sciencedirect.com/science/article/B6V4T-4MT5542-1/2/>

- b223052e4550a99626e0891d01379e9d. SAFECOMP 2004, the 23rd International Conference on Computer Safety, Reliability and Security.
- [62] R. Gulati and J.B. Dugan. A modular approach for analyzing static and dynamic fault trees. In *Reliability and Maintainability Symposium. 1997 Proceedings, Annual*, pages 57–63, 1997. doi: 10.1109/RAMS.1997.571665.
 - [63] Y. Dutuit and A. Rauzy. A linear-time algorithm to find modules of fault trees. *IEEE Transactions on Reliability*, 45(3):422–425, 1996. ISSN 0018-9529. doi: 10.1109/24.537011.
 - [64] S. Amari, G. Dill, and E. Howald. A new approach to solve dynamic fault trees. In *Reliability and Maintainability Symposium, 2003. Annual*, pages 374–379, 2003.
 - [65] M. Malhotra and K.S. Trivedi. Dependability modeling using petri-nets. *IEEE Transactions on Reliability*, 44(3):428–440, Sep 1995. ISSN 0018-9529. doi: 10.1109/24.406578.
 - [66] R. Manian, D.W. Coppit, K.J. Sullivan, and J. Bechta Dugan. Bridging the gap between systems and dynamic fault tree models. In *Reliability and Maintainability Symposium, 1999. Proceedings. Annual*, pages 105–111, 1999. doi: 10.1109/RAMS.1999.744104.
 - [67] Max Walter. Opensesame: A tool’s concept. In Carleton Scientific, editor, *Proceedings of the Satellite Workshops of the 27th International Colloquium on Automata Languages, and Programming*, volume 8, Proceedings in Informatics, 2000. doi: 10.1.1.32.3860.
 - [68] J. B. Fussel, E. F. Aber, and R. G. Rahl. On quantitative analysis of pand failure logic. *IEEE Transactions on Reliability*, R-25(5):324–326, 1976.
 - [69] W. Long, Y. Sato, and M. Horigome. Quantification of sequential failure logic for fault tree analysis. *Reliability Engineering and System Safety*, 67(3):269 – 274, 2000. ISSN 0951-8320. doi: 10.1016/S0951-8320(99)00075-7. URL <http://www.sciencedirect.com/science/article/B6V4T-3YJYP1S-6/2/89ca424aef44b6c92005d04b208b5e7b>.
 - [70] P.G. Wijayarathna and M. Maekawa. Extending fault trees with an and-then gate. In *Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on*, pages 283–292, 2000. ISBN 0-7695-0807-3. doi: 10.1109/ISSRE.2000.885879.
 - [71] J. Gorski. Extending safety analysis techniques with formal semantics. In F. J. Redmill and T. Anderson, editors, *Technology and Assessment of Safety Critical Systems*, pages 147–163. Springer-Verlag, 1994.
 - [72] J. Gorski and A. Wardzinski. Timing aspects of fault tree analysis of safety critical systems. In F. J. Redmill and T. Anderson, editors, *Safer Systems*. Springer-Verlag, 1997.
 - [73] Girish Keshav Palshikar. Temporal fault trees. *Information and Software Technology*, 44(3):137–150, 2002. ISSN 0950-5849. doi: 10.1016/S0950-5849(01)00223-3. URL <http://www.sciencedirect.com/science/article/B6V0B-44V20HW-1/2/a4d7c450faa2693d1a961c74f1ec0180>.
 - [74] Andreas Thums. *Formale Fehlerbaumanalyse*. PhD thesis, Universität Augsburg, Fakultät für Angewandte Informatik, Lehrstuhl für Softwaretechnik und Programmiersprachen, 2004.
 - [75] Antony Galton, editor. *Temporal Logics and their applications*. Academic Press, 1987.

- [76] Martin Walker and Yiannis Papadopoulos. Pandora: The time of priority-and gates. In Alexandre Dolgui, Gerard Morel, and Carlos E. Pereira, editors, *Information Control Problems in Manufacturing 2006*, pages 235–240. Elsevier Science Ltd, Oxford, 2006. ISBN 978-0-08-044654-7. doi: 10.1016/B978-008044654-7/50173-4. URL <http://www.sciencedirect.com/science/article/B87GH-4PT2PKK-3W/2/2679ad6fefcd4b7f932cbfa8a7569e96>.
- [77] Martin Walker and Yiannis Papadopoulos. PANDORA 2 : The Time of Priority-OR Gates. *IFAC Workshop on Dependable Control of Discrete Event Systems*, 2007.
- [78] Thorsten Tietjen and Dieter H. Müller. *FMEA- Praxis. Das Komplettpaket für Training und Anwendung*. Hanser Fachbuch, München, 2nd edition, 2003.
- [79] Kurt Reinschke and Igor Alekseevič Ušakov. *Zuverlässigkeitsstrukturen*. R. Oldenbourg Verlag, München, Wien, 1988.
- [80] J.A. Abraham. An improved algorithm for network reliability. *IEEE Transactions on Reliability*, R-28(1):58–61, 1979. ISSN 0018-9529. doi: 10.1109/TR.1979.5220476.
- [81] K.D. Heidtmann. Smaller sums of disjoint products by subproduct inversion. *IEEE Transactions on Reliability*, 38(3):305–311, 1989. ISSN 0018-9529. doi: 10.1109/24.44172.
- [82] R. Bertschy and P. A. Monney. A generalization of the algorithm of heidtmann to non-monotone formulas. *Journal of Computational and Applied Mathematics*, 76(1–2):55–76, 1996. ISSN 0377-0427. doi: 10.1016/S0377-0427(96)00089-1. URL <http://www.sciencedirect.com/science/article/B6TYH-3YVVXP2-4/2/30427cc15e487c973f67ef92f6cfae07>.
- [83] J. Kohlas and P. A. Monney. *A Mathematical Theory of Hints. An Approach to the Dempster-Shafer Theory of Evidence*, volume 425 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1995.
- [84] Glen B. Alleman. *Fault-Tolerant System Reliability In The Presence Of Imperfect Diagnostic Coverage*. 1989, 2000.
- [85] Simon J. Schilling. On the use of “Probabilities” in IEC 61508. *BMW Group report*, 2007.
- [86] R.M. Sinnamon and J.D. Andrews. Fault tree analysis and binary decision diagrams. In *Reliability and Maintainability Symposium, 1996 Proceedings. 'International Symposium on Product Quality and Integrity', Annual*, pages 215–222, 1996. doi: 10.1109/RAMS.1996.500665.
- [87] Formal Software Construction Ltd. OpenFTA. Cardiff, Wales, UK. URL <http://www.openfta.com>.

Appendix

A Further Explanations on Selected Topics

A.1 Reliability Characteristics

The probabilistic description of the failure behaviour of systems is done using *characteristics*, see table A.1. These are stochastic or probabilistic values, as the deterministic failure behaviour of an individual component or an individual system is usually not known in advance. Taking the probability distributions into account that result from such values is difficult in many real applications, in particular because of the effort necessary to assemble knowledge on the kind of distribution. In many cases constant or mean values are thus used instead of distributed values.

This thesis uses the terms failure probability, failure frequency and failure rate, even if it originates in a safety background, as

- the essential statements apply to the field of general reliability analogously and
- the use of such terms, that originally come from general reliability, is very common in the context of safety; see e.g. the relevant safety standards ISO 26262 [3] and IEC 61508 [4].

non repairable systems			
reliability		safety	
<i>charact.</i>	<i>symbol</i>	<i>charact.</i>	<i>symbol</i>
failure probability	$F(t)$	hazard-probability	$G(t)$
reliability	$R(t)$	safety-probability	$S(t)$
failure frequency	$f(t)$	hazard-density	$g(t)$
failure rate	$h(t)$	hazard-rate	$\delta(t)$
if constant:	λ		
repairable systems			
reliability		safety	
<i>charact.</i>	<i>symbol</i>	<i>charact.</i>	<i>symbol</i>
repair rate	$\mu(t)$	safety-restoration rate	$\nu(t)$
probability of restoration	$M(t)$	probability of safety-restoration	$W(t)$
repair frequency	$m(t)$	frequency of safety-restoration	$w(t)$
availability	$V(t)$	safety-availability	$V_S(t)$
unavailability	$U(t)$	“safety-unavailability“	$U_S(t)$

Table A.1: Characteristics of reliability and safety analysis according to [14]

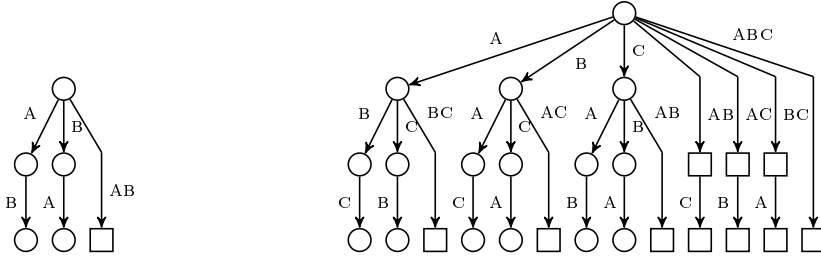
A.2 Creating and Using Sequential Failure Trees in the TFTA

Sequential failure trees allow visualization of temporal-logical expressions, as well as manual verification of transformations according to the laws of TFTA's temporal logic. Creating a sequential failure tree corresponding to a complex temporal expression requires some effort, but it is based on only a few basic steps.

Choosing the Right Failure Tree

The number of basic events within a temporal expression determines what kind of sequential failure tree needs to be chosen. The failure tree must at least support the number of basic events, but it may be bigger, too. Depending on the particular application, the simplified sequential failure tree without SAND may be sufficient.

An example: the following figure shows two sequential failure trees, that are both suited for the expression $\varpi = A \wedge B$ and are not yet filled in.



Transforming the Temporal Expression

If the temporal expression is too complex, then, in a first step, simple sub-expressions need to be identified, and for these sequential fault trees are then created. As an extreme example, the basic events of the temporal expression are chosen. The following steps are then repeated for all these sub-expressions.

For instance, the two sub-expressions A and B are chosen for the expression $\varpi = A \wedge B$.

Minimal Failure Nodes

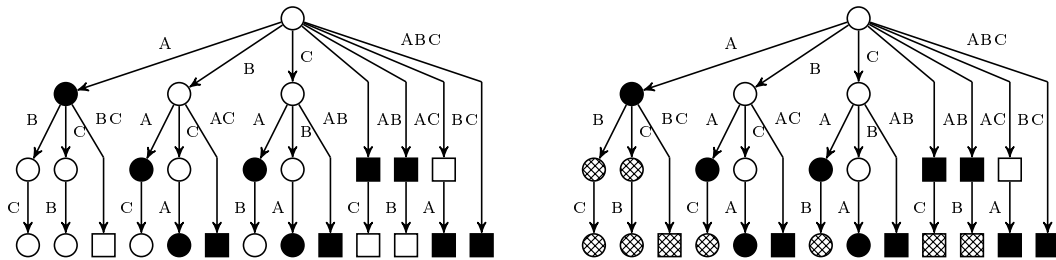
Starting with the top-node all branches of the sequential failure tree are walked along, until in each branch the currently chosen sub-expression has occurred (or the branch has ended), and the minimal failure nodes are tagged.

An example is presented in the next step.

Non-Minimal Failure Nodes

All nodes beneath a minimal failure node are tagged as successor nodes.

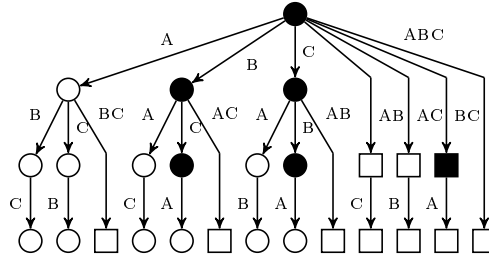
An example: the following figures show the minimal (on the left side) as well as minimal and successor failure nodes (right side) corresponding to the temporal expression $\varpi = A$.



Negated Events

Starting with the sequential failure tree corresponding to an event, all original non-failure nodes are marked as new minimal failure nodes; and all original failure nodes (minimal as well as successor) are marked as non-failure nodes. *No new* non-minimal failure nodes are added.

The following figure shows the sequential failure tree for the example of $\neg A$.



Conjunction/AND Relationship

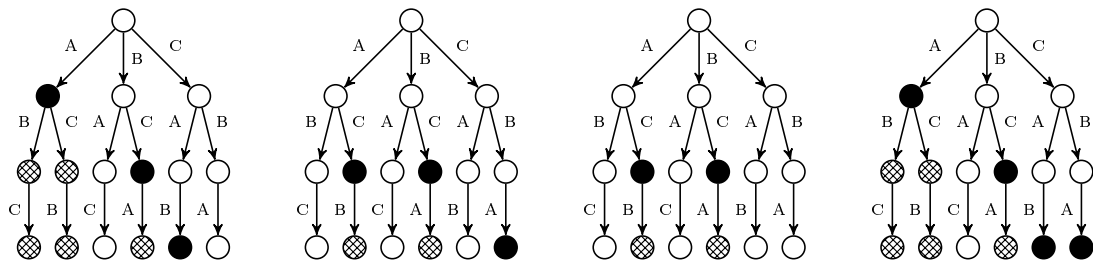
The sequential failure tree of the conjunction of two temporal expressions is the “intersection” of the individual expressions’ sequential failure trees. Minimal failure nodes thereby absorb non-minimal failure nodes. In a next step, non-minimal failure nodes are added as necessary; this is especially necessary in case of negated events.

An example is presented in the next step.

Disjunction/OR Relationship

The sequential failure tree of the disjunction of two temporal expressions is the “union” of the individual expressions’ sequential failure trees. Non-minimal failure nodes thereby absorb minimal failure nodes. In a next step, non-minimal failure nodes are added as necessary; this is especially necessary in case of negated events.

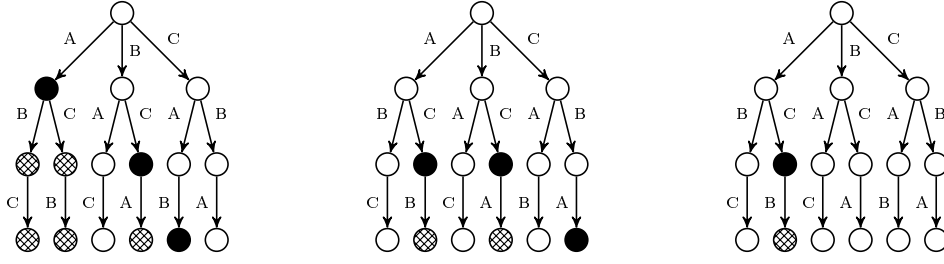
An example: the following figure shows (from left to right) two simplified sequential failure trees, as well as their “intersection” and “union”, respectively.



PAND Relationship

The sequential failure tree of the PAND connection of two temporal expressions, i.e. $\varpi_1 \vec{\wedge} \varpi_2$, is generated as follows: All those nodes are marked as minimal failure nodes that are minimal failure nodes of ϖ_2 together with being non-minimal failure nodes of ϖ_1 . In a next step, non-minimal failure nodes are added as necessary.

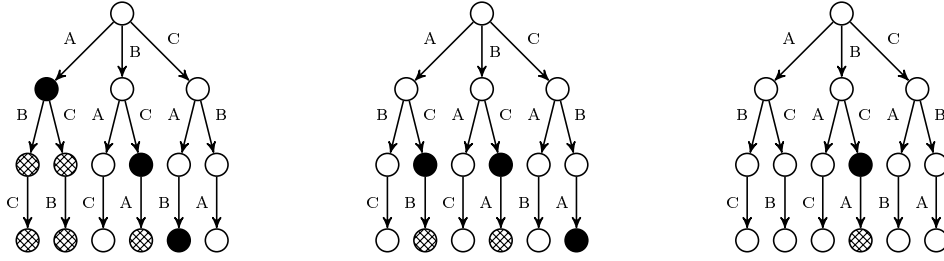
An example: the following figure shows (from left to right) two simplified sequential failure trees and their PAND connection.



SAND Relationship

The sequential failure tree of the SAND connection of two temporal expressions, i.e. $\varpi_1 \bar{\wedge} \varpi_2$, is generated as follows: All those nodes are marked as minimal failure nodes that are minimal failure nodes of ϖ_2 together with being minimal failure nodes of ϖ_1 . In a next step, non-minimal failure nodes are added as necessary.

An example: the following figure shows (from left to right) two simplified sequential failure trees and their SAND connection.



A.3 Examples: Mutually Exclusive (Disjoint) Temporal Expressions

The following assumes $n = 3$ and failure events A , B , and C .

First Example

The failure function $\varpi = B$ is already given as a TDNF with only one sub-expression; it is not a minterm, though, as not all possible failure events are included in this expression. Using the method provided on page 52 yields a TDNF of mutually exclusive (disjoint) and minimal event sequences, that are temporal minterms, too:

$$\begin{aligned} B &= B \wedge (\neg A \vee A) \wedge (\neg C \vee C) = \\ &= [A \wedge B \wedge C] \vee [\neg C \wedge (A \wedge B)] \vee [\neg A \wedge (B \wedge C)] \vee [(\neg A \neg C) \wedge B] . \end{aligned}$$

For better readability, the four resulting sub-expressions are inspected separately.

$$\eta_1 = A \wedge B \wedge C .$$

Using the law of completion twice yields

$$\begin{aligned} \eta_1 &= [(A \wedge B) \bar{\wedge} C] \vee [(A \wedge B) \bar{\wedge} \bar{C}] \vee [C \bar{\wedge} (A \wedge B)] = \\ &= [(A \bar{\wedge} B \vee A \bar{\wedge} \bar{B} \vee B \bar{\wedge} A) \bar{\wedge} C] \vee [(A \bar{\wedge} B \vee A \bar{\wedge} \bar{B} \vee B \bar{\wedge} A) \bar{\wedge} \bar{C}] \vee \\ &\quad \vee [C \bar{\wedge} (A \bar{\wedge} B \vee A \bar{\wedge} \bar{B} \vee B \bar{\wedge} A)] . \end{aligned}$$

As the expressions in round brackets are mutually exclusive (disjoint),

$$\eta_1 = [A \bar{\lambda} B \bar{\lambda} C] \vee [(A \bar{\lambda} B) \bar{\lambda} C] \vee [B \bar{\lambda} A \bar{\lambda} C] \vee [(A \bar{\lambda} B) \bar{\lambda} C] \vee [(A \bar{\lambda} B) \bar{\lambda} C] \vee \\ \vee [(B \bar{\lambda} A) \bar{\lambda} C] \vee [C \bar{\lambda} (A \bar{\lambda} B)] \vee [C \bar{\lambda} (A \bar{\lambda} B)] \vee [C \bar{\lambda} (B \bar{\lambda} A)] .$$

Applying the transformation laws of the temporal logic then yields

$$\eta_1 = [A \bar{\lambda} B \bar{\lambda} C] \vee [(A \bar{\lambda} B) \bar{\lambda} C] \vee [B \bar{\lambda} A \bar{\lambda} C] \vee [A \bar{\lambda} (B \bar{\lambda} C)] \vee [A \bar{\lambda} B \bar{\lambda} C] \vee \\ \vee [B \bar{\lambda} (A \bar{\lambda} C)] \vee [A \bar{\lambda} C \bar{\lambda} B] \vee [(A \bar{\lambda} C) \bar{\lambda} B] \vee [C \bar{\lambda} A \bar{\lambda} B] \vee \\ \vee [B \bar{\lambda} C \bar{\lambda} A] \vee [(B \bar{\lambda} C) \bar{\lambda} A] \vee [C \bar{\lambda} B \bar{\lambda} A] \vee [C \bar{\lambda} (A \bar{\lambda} B)] .$$

With this the transformation of the first sub-expression is completed.

Now, applying the law of completion on the second sub-expression, i.e.

$$\eta_2 = \neg C \wedge (A \wedge B)$$

yields already disjoint expressions, thus

$$\eta_2 = \neg C \wedge ((A \bar{\lambda} B \vee (A \bar{\lambda} B) \vee (B \bar{\lambda} A)) = \\ = [\neg C \wedge (A \bar{\lambda} B)] \vee [\neg C \wedge (A \bar{\lambda} B)] \vee [\neg C \wedge (B \bar{\lambda} A)] .$$

The third sub-expression is transformed analogously, thus

$$\eta_3 = \neg A \wedge (B \wedge C) = [\neg A \wedge (B \bar{\lambda} C)] \vee [\neg A \wedge (B \bar{\lambda} C)] \vee [\neg A \wedge (C \bar{\lambda} B)] .$$

The fourth sub-expression consists of one event sequence, that cannot be further simplified:

$$\eta_4 = (\neg A \neg C) \wedge B .$$

Combining these results, the three-variables minterm form of expression $\varpi = B$ is given as (meaning of underlines, see below):

$$\varpi = B = \eta_1 \vee \eta_2 \vee \eta_3 \vee \eta_4 = \\ = [\underline{A \bar{\lambda} B \bar{\lambda} C}] \vee [\underline{(A \bar{\lambda} B) \bar{\lambda} C}] \vee [\underline{B \bar{\lambda} A \bar{\lambda} C}] \vee [A \bar{\lambda} (B \bar{\lambda} C)] \vee [A \bar{\lambda} B \bar{\lambda} C] \vee \\ \vee [B \bar{\lambda} (A \bar{\lambda} C)] \vee [A \bar{\lambda} C \bar{\lambda} B] \vee [(A \bar{\lambda} C) \bar{\lambda} B] \vee [C \bar{\lambda} A \bar{\lambda} B] \vee [\underline{B \bar{\lambda} C \bar{\lambda} A}] \vee \\ \vee [(B \bar{\lambda} C) \bar{\lambda} A] \vee [\underline{C \bar{\lambda} B \bar{\lambda} A}] \vee [\neg C \wedge (A \bar{\lambda} B)] \vee [\neg C \wedge (A \bar{\lambda} B)] \vee \\ \vee [\neg C \wedge (B \bar{\lambda} A)] \vee [\neg A \wedge (B \bar{\lambda} C)] \vee [\neg A \wedge (B \bar{\lambda} C)] \vee [\neg A \wedge (C \bar{\lambda} B)] \vee \\ \vee [C \bar{\lambda} (A \bar{\lambda} B)] \vee [(\neg A \neg C) \wedge B] .$$

In this form ϖ is not yet minimal. As shown in figure A.1, only eleven of the 20 nodes, in which $B = \text{True}$, are really minimal. The minterms corresponding to these non-minimal nodes are underlined in the figure above. Applying the temporal laws of absorption provides the following minimal form, where

$$\varpi = B = \eta_1 \vee \eta_2 \vee \eta_3 \vee \eta_4 = \\ = [A \bar{\lambda} (B \bar{\lambda} C)] \vee [A \bar{\lambda} B \bar{\lambda} C] \vee [A \bar{\lambda} C \bar{\lambda} B] \vee [(A \bar{\lambda} C) \bar{\lambda} B] \vee [C \bar{\lambda} A \bar{\lambda} B] \vee \\ \vee [\neg C \wedge (A \bar{\lambda} B)] \vee [\neg C \wedge (A \bar{\lambda} B)] \vee [\neg A \wedge (B \bar{\lambda} C)] \vee \\ \vee [\neg A \wedge (C \bar{\lambda} B)] \vee [C \bar{\lambda} (A \bar{\lambda} B)] \vee [(\neg A \neg C) \wedge B] .$$

Specifically, the structurally and temporally non-minimal temporal expressions (see chapter 4.3.2) demonstrate that

$(\neg A \neg C) \wedge B$	covers $[B \vec{\wedge} (A \bar{\wedge} C)]$, $[\neg A \wedge (B \vec{\wedge} C)]$, $[\neg C \wedge (B \vec{\wedge} A)]$,
$\neg A \wedge (B \vec{\wedge} C)$	covers $B \vec{\wedge} C \vec{\wedge} A$,
$\neg C \wedge (B \vec{\wedge} A)$	covers $B \vec{\wedge} A \vec{\wedge} C$,
$\neg C \wedge (A \vec{\wedge} B)$	covers $A \vec{\wedge} B \vec{\wedge} C$,
$\neg A \wedge (C \vec{\wedge} B)$	covers $C \vec{\wedge} B \vec{\wedge} A$,
$\neg C \wedge (A \bar{\wedge} B)$	covers $(A \bar{\wedge} B) \vec{\wedge} C$,
$\neg A \wedge (B \bar{\wedge} C)$	covers $(B \bar{\wedge} C) \vec{\wedge} A$.

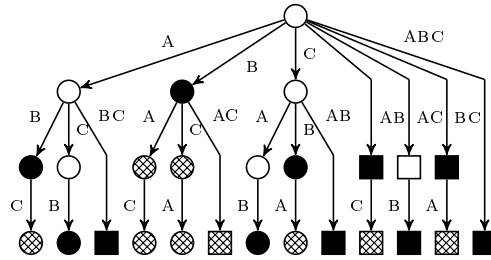


Figure A.1: Sequential failure tree corresponding to $\varpi = B$ with eleven minimal failure nodes and nine non-minimal failure nodes. Nine failure nodes also include at least one SAND connection.

Second Example

The failure function $\varpi = (A \vee B) \vec{\wedge} C$ is not presented in a TDNF. First, the transformation laws of temporal logic are used in order to create a TDNF:

$$\varpi = (A \vee B) \vec{\wedge} C = (A \vec{\wedge} C) \vee (B \vec{\wedge} C) .$$

Both sub-expressions on the right side do not include all three relevant variables. Each sub-expression is therefore transformed according to (4.122) as to include the missing variables.

$$\begin{aligned} \varpi &= [\neg B \wedge (A \vec{\wedge} C)] \vee [B \wedge (A \vec{\wedge} C)] \vee [\neg A \wedge (B \vec{\wedge} C)] \vee [A \wedge (B \vec{\wedge} C)] = \\ &= [\neg B \wedge (A \vec{\wedge} C)] \vee [B \vec{\wedge} A \vec{\wedge} C] \vee [A \vec{\wedge} B \vec{\wedge} C] \vee [(A \bar{\wedge} B) \vec{\wedge} C] \vee \\ &\quad \vee [A \vec{\wedge} (B \bar{\wedge} C)] \vee [A \vec{\wedge} C \vec{\wedge} B] \vee [\neg A \wedge (B \vec{\wedge} C)] \vee [A \vec{\wedge} B \vec{\wedge} C] \vee \\ &\quad \vee [B \vec{\wedge} A \vec{\wedge} C] \vee [(A \bar{\wedge} B) \vec{\wedge} C] \vee [B \vec{\wedge} (A \bar{\wedge} C)] \vee [B \vec{\wedge} C \vec{\wedge} A] . \end{aligned}$$

The expressions $A \vec{\wedge} B \vec{\wedge} C$ and $B \vec{\wedge} A \vec{\wedge} C$ and $(A \bar{\wedge} B) \vec{\wedge} C$ are listed twice each. Moreover, $\neg A \wedge (B \vec{\wedge} C)$ and $\neg B \wedge (A \vec{\wedge} C)$ cover the non-minimal expressions $B \vec{\wedge} C \vec{\wedge} A$ und $A \vec{\wedge} C \vec{\wedge} B$. Thus, the minterm-form of the failure function is given as

$$\begin{aligned} \varpi &= [\neg B \wedge (A \vec{\wedge} C)] \vee [B \vec{\wedge} A \vec{\wedge} C] \vee [A \vec{\wedge} B \vec{\wedge} C] \vee [(A \bar{\wedge} B) \vec{\wedge} C] \vee \\ &\quad \vee [A \vec{\wedge} (B \bar{\wedge} C)] \vee [\neg A \wedge (B \vec{\wedge} C)] \vee [B \vec{\wedge} (A \bar{\wedge} C)] . \end{aligned}$$

Figure A.2 shows the sequential failure tree of this second example, including its seven minimal and two non-minimal failure nodes.

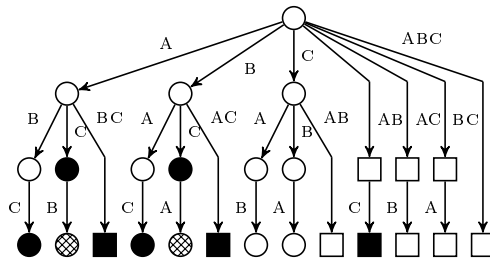


Figure A.2: Sequential failure tree corresponding to $\varpi = (A \vee B) \vec{\wedge} C$ seven minimal and two non-minimal failure nodes. Three failure nodes include at least one SAND connection.

B Abbreviations/Acronyms

BDD	binary decision diagram	FTA	fault tree analysis
BDMP	Boolean logic driven markov processes	HRA	human reliability analysis
CCF	common cause failure	MoCaS	monte-carlo-simulation
DFT	dynamic fault tree	MCSS	minimal cutset sequences
DGL	differential equation	PAND	priority AND
DNF	disjunctive normal form	POR	priority OR
DRBD	dynamic reliability block diagram	RBD	reliability block diagram
E/E	electric/electronic	SAND	simultaneous AND
FAA	federal aviation administration	TDNF	temporal disjunctive normal form
FMEA	failure modes and effects analysis	TFTA	temporal fault tree analysis
FT	fault tree	ZSA	reliability and safety analyses

C Notation

Symbol	Meaning
$\cdot(t)$	time dependent parameter \cdot
\cdot_i	parameter \cdot for element i
$o(\Delta t)$	function with $\lim_{\Delta t \rightarrow 0} \frac{o(\Delta t)}{\Delta t} = 0$
\wedge	Boolean AND
\vee	Boolean OR
\neg	Boolean NOT
$\vec{\wedge}$	temporal PAND
$\overline{\wedge}$	temporal SAND
$\subset; \subseteq$	proper subset; subset
\perp	are disjoint (for events, e.g. $A \vec{\wedge} B \perp B \vec{\wedge} A$)
\in	is element of (for sets, e.g. $1 \in \{1, 2, \dots, n\}$)
\in	is part of (for events, e.g. $A \in A \vec{\wedge} B$)
\exists	there is
\nsubseteq	is minimal
A, B, C, D	failure events (within examples), see X
ae	token for atomic events
ce	token for core events
E	expectancy value
eK	extended core event
ece	token for extended core events
ES	event sequence
es	token for event sequences
eES	extended event sequence
ees	token for extended event sequences
etdnf	token for extended temporal expressions in TDNF
η	temporal (sub)expression (in chapter 7 and appendix A)
f	failure density (density function of the failure probability)
F	failure probability/unavailability
i	index
j	index
k	index
k	position of an extended core event within an extended MCSS
K	core event
\vec{K}	system-state-vector/-node (sequential failure tree)
\vec{K}'	predecessor node (sequential failure tree)
\vec{K}''	successor node (sequential failure tree)

continued on next page

continued

Symbol	Meaning
l	index
λ	failure rate
$\lambda_{i,j}$	transition rate between states i and j
$\max(.)$	maximum
MS	minimal cutset
$MCSS$	minimal cutset sequence
n	index
nae	token for negated atomic events
nce	token for negated core events
nes	token for event sequences with negated events
$nees$	token for extended event sequences with negated events
$O\{x\}$	order of complexity x
P	state probability
\dot{P}	derivative of the state probability
φ	Boolean failure function
ϖ	temporal failure function
r	system state (sequential failure tree)
r	number of AND-connected basic events within an extended core event
R	reliability
S	cutset (as in minimal cutset)
t	time
t_X	time of occurrence of event X (at this time the failure represented by X occurs)
T	life expectancy
T_M	mission time
τ	time (parameter in integrations)
$\tau^{\{i\}}$	i -th parameter in integrations in multiple integrals
Δt	(infinitesimally) small time span
$tdnf$	token for temporal expressions in TDNF
u	index
U	unavailability
w	number of extended core events within an extended MCSS
X	Boolean event (failure logic: $X = 1 \rightarrow$ failed, $X = 0 \rightarrow$ not failed)
γ	number of MCSS covered by an extended MCSS
ζ	number of cutsets
ξ	number of minimal cutsets